MinMax Recurrent Neural Cascades

Alessandro Ronca
IRIS-AI, Montefiascone, Italy
alessandro.ronca@iris-ai.org

Abstract

We introduce *MinMax Recurrent Neural Cascades (MinMax RNCs)*, a class of recurrent neural networks based on a novel form of recurrence in the MinMax algebra. MinMax RNCs are a first in simultaneously enjoying the key properties for truly long-context processing: strong formal expressivity with long-term memory, efficient inference both in recurrent and parallel form, stable dynamics and gradients, and long-range sensitivity ensured by non-vanishing state gradients. First, they can store and retain information arbitrarily long, and manipulate it in a rich way, with a formal expressivity corresponding the regular languages—arguably the maximal expressivity for finite-memory systems. Second, in addition to linear-time inference via recurrence, they also admit log-depth–linear-work inference via parallel scan. Third, their states and activations are uniformly bounded independently of sequence length. Fourth, although not everywhere-differentiable, they admit pseudo-gradients that are uniformly bounded independently of sequence length and coincide with ordinary gradients almost everywhere. Fifth, they do not suffer from vanishing state gradients: the pseudo-gradient of a state with respect to an arbitrarily-distant past state can retain unit norm, while coinciding with the ordinary gradient. Empirically, we find that the theoretical properties translate to strong practical performance. We evaluate MinMax RNCs on a suite of fundamental synthetic tasks, and observe they solve all tasks perfectly, generalising seamlessly on longer sequences when tested up to length 1M, and outperforming a selection of baselines that well-represent the state of the art. We train a 112M-parameter MinMax RNC for next-token prediction in OWT, obtaining competitive performance for its size, and hence initial evidence of the potential of MinMax RNCs for real-world sequence-modelling tasks.

1 Introduction

Sequence models face a tension between computational efficiency, expressivity, stability, and long-range sensitivity. *Transformers* [1] enable highly parallel computation, but standard full self-attention has quadratic cost in the sequence length, and it is unable to detect basic patterns such as interleaving of designated tokens, as it is insensitive to inputs occurring early on [2]. *Nonlinear recurrence* as in [3–6] avoids the quadratic cost and yields an expressivity including all regular languages [7], but it is inherently sequential and can suffer from vanishing state gradients [8].

Linear recurrence underlies a broad range of recent architectures [9–30] and offers a tradeoff: linear state updates compose associatively and therefore admit evaluation via *parallel scan* [31], but they introduce a challenge in ensuring stability. A common solution is to adopt *contractive parametrisations*—as in Mamba [14], mLSTM [6], xLSTM-L [32], Gated DeltaNet [28], and Gated DeltaProduct [25]—which ensure stability but introduce their own limitations. Although existing results show that models like Mamba have an expressivity including all star-free regular languages, such results rely on decoding outputs with a separation margin that vanishes with sequence length. We

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

introduce a notion of *positive-margin expressivity* that requires outputs to have a length-independent positive separation margin, and show that all contractive Linear RNNs have a positive-margin expressivity within the so-called *definite languages*, reflecting the fact that sufficiently long suffixes erase the influence of earlier prefixes (see §R, §T, §U). Abandoning contraction avoids the expressivity limitations, as states can propagate with no attenuation. However, it generally results into instability: *expansive parametrisations* as in RWKV-7 [18] are unstable as they can amplify states at each update step (cf. [25]); and even *non-expansive parametrisations* as in DeltaNet [24, 33] and its generalisation DeltaProduct [25] are unstable as their additive inputs can accumulate across state updates yielding divergent state trajectories, and also linear growth of loss gradients for increasing sequence length (see §S). What is needed is therefore a mechanism that can propagate states without attenuation, amplification, or additive accumulation.

MinMax recurrence provides such a mechanism. It is the affine recurrence in the MinMax algebra, obtained from ordinary affine recurrence by replacing sum and product $(+, \times)$ with their MinMax counterparts $(\oplus, \otimes)$ that use max/min in place of addition/multiplication at the level of scalars,

$$x_t = (A_t \otimes x_{t-1}) \oplus b_t \qquad \text{for} \quad A_t \in \mathbb{R}^{d\times d}, \;\; b_t, x_t \in \mathbb{R}^d .$$

MinMax recurrence retains the algebraic properties that enable parallel scan, while yielding the desired form of state propagation since min/max operations simply select coordinates among the ones of the previous state x_{t-1} and of the inputs $\langle A_t, b_t\rangle$, thus avoiding the above issues. Additionally, unlike ordinary affine recurrences, MinMax recurrence is numerically exact in floating-point arithmetic as its updates involve only min and max, further ensuring stability of implementations on long sequences.

Overview. We present *MinMax Recurrent Neural Cascades (MinMax RNCs)*, that adopt a general architecture obtained by cascading layers of *MinMax Neurons*, that employ MinMax recurrence with feedforward networks $\langle R, s\rangle$ yielding $A_t = R(u_t)$ and $b_t = s(u_t)$ on input u_t. We show they enjoy the desired theoretical properties jointly, and we assess them experimentally observing strong performance in line with the theoretical analyses, hence obtaining an experimental validation of the faithfulness of such analyses in characterising the practical capabilities of MinMax RNCs.

- *Expressivity (§4).* MinMax Neurons can realise all fundamental semiautomata (Theorems 3, 4), making MinMax RNCs capable of recognising all regular languages, also when requiring a positive margin on output decoding (Theorem 1). The state dimension of a MinMax Neuron corresponds to the *permutation degree* of the semiautomata it can realise. Thus, scalar states suffice for all star-free languages, and d-dimensional states suffice for all languages whose canonical automaton admits a cascade decomposition into semiautomata of permutation degree at most d.
- *Complexity (§5).* The computational complexity of MinMax RNCs stems from the core task of evaluating MinMax recurrence. For T the sequence length and d the state dimension, MinMax recurrence can be evaluated sequentially in time $O(Td^2)$ (Theorem 5), and it admits an associative recurrence-composition operator enabling parallel-scan evaluation with depth $O(\log T \log d)$ and work $O(Td^3)$, hence runtime $O(\log T \log d)$ given $\Theta(T)$ processors (Theorem 5).
- *Stability and gradients (§6).* MinMax RNCs have states, activations, and outputs bounded uniformly on bounded inputs and parameters independently of sequence length (Theorem 7). Although they are not everywhere-differentiable due to min/max operations, we identity a pseudo-derivative operator that is in line with PyTorch's AutoDiff [34] and coincides with ordinary differentiation almost everywhere; the corresponding state pseudo-Jacobians and loss pseudo-gradients are bounded uniformly on bounded inputs and parameters independently of sequence length (Theorem 8). Furthermore, all MinMax Neurons underlying the expressivity of MinMax RNCs admit input sequences over which their state pseudo-Jacobians retain norm one at all timesteps while coinciding with the ordinary Jacobian eveywhere (Theorem 9).

  On way to see how MinMax recurrence is able to propagate states with no attenuation while ensuring stability is to note that its state updates are *piecewise affine* with a special form (§N). The state-input space admits a finite partition into regions Z with affine state update

$$x_t = A_Z \cdot x_{t-1} + b_Z(u_t) \qquad \text{with} \quad (A_Z)_{i,:} = \mathbf{0} \quad \text{or} \quad (A_Z)_{i,:} = \mathbf{e}_\ell, \; (b_Z)_i = 0 ,$$

  for $\mathbf{e}_\ell$ a standard basis row-vector. Thus, the state updates are non-expansive since $\|A_Z\| \leq 1$, with $\|A_Z\| = 1$ when A_Z is a permutation matrix, with identity as a special case; at the same time, $(b_Z)_i$ being identically zero whenever $(A_Z)_{i,:}$ is nonzero ensures no additive accumulation.

- *Empirical evaluation on star-free tasks (§7.1).* We evaluate MinMax RNCs with scalar-state neurons, on a suite of star-free tasks, comparing it against the following baselines: Mamba [14], mLSTM [6], and xLSTM-L [32], based on contractive linear recurrence; DeltaNet [24, 33] based on non-expansive linear recurrence; sLSTM [6] based on nonlinear recurrence. MinMax RNCs achieve perfect validation accuracy on all tasks, retaining the same accuracy when evaluated on longer sequences up to the considered maximum length 1M. In contrast, all baselines show an evaluation accuracy that decreases with sequence length, dropping to random guess already at length 100–200k. MinMax RNCs have competitive throughput compared to the baselines.
- *Empirical evaluation on tasks beyond star-free (§7.2).* We evaluate MinMax RNCs on the *word problems* of the groups S_3 and S_4. They achieve perfect validation accuracy, and retaining the same accuracy on the considered max length 1M, in contrast with DeltaProduct whose accuracy begins to deteriorate already at length 200–500 [25].
- *Empirical evaluation on next-token prediction (§7.3).* We train a MinMax RNC of 112M parameters on next-token prediction in OpenWebText (OWT), achieving a loss comparable to GPT-2, and hence gaining evidence of the potential of MinMax RNCs in real-world applications.

The paper presents the results along with the key supporting arguments. The appendices provide extended background (§A), auxiliary notions and results (§B–§H), full proofs (§I–§N), details of the implementation (§O) and of the experiments (§P–§Q), technical analysis of related work (§R–§U).

2 Preliminaries

For $i, j \in \mathbb{N}$ with $i \leq j$, we define $[i..j] := \{i, i+1, \ldots, j\}$ and $[j] := [1..j]$. For X a set, X^* is the set of all non-empty finite sequences of elements of X, and X^+ is X^* excluding the empty sequence. For X a normed space and $r \in \mathbb{R}^{>0}$, $X_{\leq r}$ is the restriction of X to elements having norm at most r.

MinMax algebra. For $a, b \in \mathbb{R}$, we define the operators $a \oplus b := \max(a, b)$ and $a \odot b := \min(a, b)$; they apply component-wise to vectors and matrices; they are associative, commutative, and $\odot$ distributes over $\oplus$. For $A \in \mathbb{R}^{m \times p}, B \in \mathbb{R}^{p \times n}$, the operator $A \otimes B$ yields $C \in \mathbb{R}^{m \times n}$ with $C_{i,j} = \bigoplus_{k=1}^{p} A_{i,k} \odot B_{k,j}$; it is associative and distributes over $\oplus$; $B \in \mathbb{R}^n$ is seen as $B \in \mathbb{R}^{n \times 1}$.

Dynamical systems. A *dynamical system* is a tuple $S = \langle X, U, f, x_0, Y, h \rangle$ where X, U, Y are metric spaces called the *state*, *input*, and *output space*, respectively; $f : X \times U \to X$ and $h : X \times U \times X \to Y$ are continuous functions called the *dynamics* and *output function*, respectively; and $x_0 \in X$ is the *initial state*. We refer to $D = \langle X, U, f \rangle$ as *dynamics*, of S in this case. An *input sequence* is $u = (u_t)_{t=1}^T \in U^+$. The *state sequence* of D on u from state x_0 is $(x_t)_{t=1}^T$ with *step-t state* $x_t := f(x_{t-1}, u_t)$, and the *output sequence* of S is $(y_t)_{t=1}^T$ with *step-t output* $y_t := h(x_{t-1}, u_t, x_t)$, with associated maps $\check{D}(x_0, u) := x_T$, $\check{D}_t^u(x_0) := x_t$, and $\check{S}(u) := y_T$.

Compositions. Given systems $S_1 = \langle X_1, U, f_1, \dot{x}_1, Z, h_1 \rangle$ and $S_2 = \langle X_2, Z, f_2, \dot{x}_2, Y, h_2 \rangle$, their *cascade composition* $S_1 \ltimes S_2$ is the system $\langle X, U, f, \dot{x}, Y, h \rangle$ having state space $X := X_1 \times X_2$, initial state $\dot{x} := \langle \dot{x}_1, \dot{x}_2 \rangle$, dynamics function $f(\langle x_1, x_2 \rangle, u) := \langle x_1', f_2(x_2, h_1(x_1, u, x_1')) \rangle$ for $x_1' = f_1(x_1, u)$, and output function $h(\langle x_1, x_2 \rangle, u, \langle x_1', x_2' \rangle) = h_2(x_2, h_1(x_1, u, x_1'), x_2')$. Similarly, dynamics can be cascaded as $D_1 \ltimes D_2$ by seeing D_1 as having an identity input function, and they can be composed in parallel as $D_1 \parallel D_2$ by seeing D_1 as having a void output function.

Algebraic Automata Theory (AAT), cf. [35, 36]. An *automaton* is a system $A = \langle Q, \Sigma, \delta, q_0, \Gamma, \theta \rangle$ where Q, Σ, Γ are finite spaces under the discrete metric, with Σ and Γ called *alphabets* and $\langle Q, \Sigma, \delta \rangle$ called a *semiautomaton*. For $\sigma \in \Sigma$, the *state transformation* δ_σ of D is $\delta_\sigma(q) = \delta(q, \sigma)$. Semiautomaton D is an *identity-reset semiautomaton* if each transformation δ_σ is identity or constant, and it is a *permutation semiautomaton* if each δ_σ is a bijection. A semiautomaton D *realises* a semiautomaton D' when there is a way to assign the states and inputs of D' to the ones of D in a consistent way; the realisability relation is transitive and closed under cascading. By the *Krohn-Rhodes theorem*, every semiautomaton D can be realised by a cascade $D_1 \ltimes \cdots \ltimes D_n$ where each D_i is an identity-reset or permutation semiautomaton closely connected to D. A semiautomaton is *group-free* if realised by a cascade of identity-reset semiautomata. A function $F : \Sigma^+ \to \Gamma$ over alphabets is *(group-free) regular* if it is defined by some (group-free) automaton. A *language* is a subset of Σ^+, and it is *(star-free) regular* if its characteristic function is (group-free) regular.

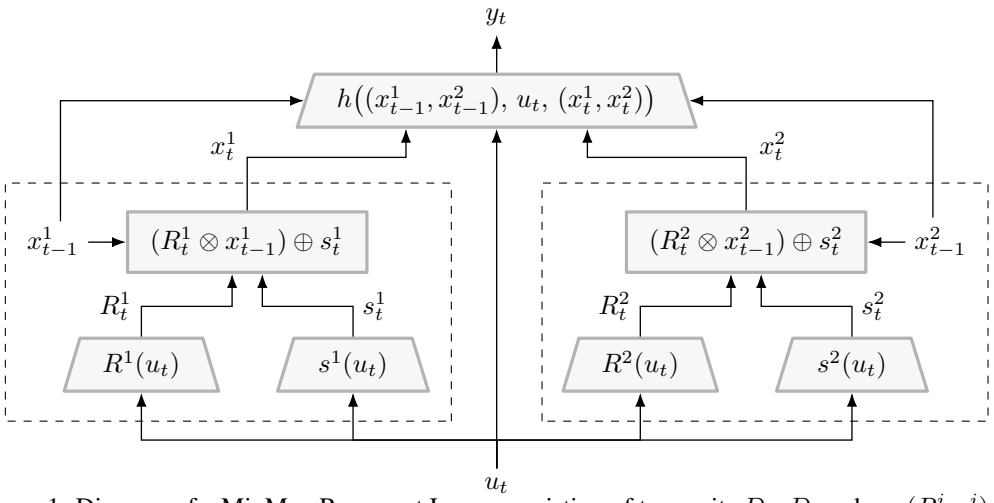


Figure 1: Diagram of a MinMax Recurrent Layer consisting of two units D_1, D_2, where (R^i, s^i) are the input functions of D_i, and h is the output function of the layer. The state (x^1_t, x^2_t) and output y_t at time t are computed from the input u_t at time t and the previous state (x^1_{t-1}, x^2_{t-1}).

Metric Automata Theory (MAT) [37]. To relate a system S to a function $F : \Sigma^+ \to \Gamma$ over alphabets, one says that S *can-implement* F if there are a continuous encoder enc and decoder dec such that $F(w) = \mathrm{dec}(S(\mathrm{enc}(w)))$ with enc applied element-wise. Automata theory can be used to establish whether S can implement F whenever S is η-finite. System S is *η-finite* if so are its input and state space. For our purposes, a space $X \subseteq \mathbb{R}$ is *η-finite* if it is the union of a finite number of compact disjoint intervals, called *η-components*, and every space $X_1 \times \cdots \times X_n$ is η-finite if so is each $X_i \subseteq \mathbb{R}$. Function F can be implemented by an η-finite system S iff F can be implemented by an automaton whose semiautomaton is realised by the *η-canonical semiautomaton* of the dynamics of S, obtained from the dynamics of S by seeing each input and state η-component as a single element. A system S *can-recognise* a language language if it can-implement its characteristic function.

3 Model

We define MinMax Recurrent Neurons as a specialisation of MinMax Recurrent Units, which formalise arbitrary dynamics based on MinMax recurrence. To favour simplicity, here we characterise Neurons in terms of MLPs [38], but in fact all our results hold under a more general characterisation that covers most feedforward neural networks—see §A.10 and §H.2.

Definition 1. We define a *MinMax Recurrent Unit* with input functions $R : U \subseteq \mathbb{R}^m \to \mathbb{R}^{n\times n}$ and $s : U \subseteq \mathbb{R}^m \to \mathbb{R}^n$ as dynamics $D = \langle X, U, f\rangle$ having state space $X \subseteq \mathbb{R}^n$ and dynamics function:

$$f(x, u) = \big(R(u) \otimes x\big) \oplus s(u).$$

We refer to R as the *reset function*, to s as the *set function*, and to each of them as an *input function*, also writing $D = \langle X, U, f \mid R, s\rangle$. Then, unit D is a *MinMax Recurrent Neuron* if $\langle R, s\rangle$ are MLPs. ∎

We present two examples to provide intuition on the mechanics of MinMax Neurons.

Example 1. For the MinMax Neuron $D = \langle \mathbb{R}, \mathbb{R}^2, f \mid R, s\rangle$ having reset function $R(a, b) = a$ and set function $s(a, b) = b$, for an initial state $x_0 = 0$ and input sequence $u = (0, 2), (7, 0), (5, 0), (0, 1)$, if we write $u_{t,1}, u_{t,2}$ for the first and second input component at time t, the state sequence is:

$$x_1 = (u_{1,1} \otimes x_0) \oplus u_{1,2} = (0 \odot 0) \oplus 2 = 2, \quad x_2 = (u_{2,1} \otimes x_1) \oplus u_{2,2} = (7 \odot 2) \oplus 0 = 2,$$

$$x_3 = (u_{3,1} \otimes x_2) \oplus u_{3,2} = (5 \odot 2) \oplus 0 = 2, \quad x_4 = (u_{4,1} \otimes x_3) \oplus u_{4,2} = (0 \odot 2) \oplus 1 = 1.$$

At step t, the neuron stores $s(u_t)$ if $R(u_t) \leq s(u_t)$, and keeps x_{t-1} if $s(u_t) \leq x_{t-1} \leq R(u_t)$. ∎

Example 2. Let $D = \langle \mathbb{R}^2, \mathbb{R}, f \mid R, s\rangle$ be the MinMax Neuron with $R(u) = \left(\begin{smallmatrix} 1-u & u \\ u & 1-u \end{smallmatrix}\right)$, $s(u) = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$. Note that $R(0) = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ is the identity matrix, and $R(1) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ the swap matrix. From $x_0 = \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)$, on the input sequence $u = 1, 0, 1$, the state sequence is $x_1 = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right), x_2 = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right), x_3 = \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)$, obtained as:

$$x_0 \;\rightsquigarrow\; \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) \oplus \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right) = x_1 \;\rightsquigarrow\; \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right) \oplus \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right) = x_2 \;\rightsquigarrow\; \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right) \oplus \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right) = x_3.$$

State components are swapped on input 1 and left unchanged on 0. Hence D implements parity. ∎

MinMax RNCs are obtained by cascading layers of MinMax Recurrent Neurons. The diagram of a two-unit layer is shown in Figure 1, and cascading corresponds to stacking such diagrams.

Definition 2. A *MinMax Recurrent Cascade* is a cascade composition $S = S_1 \ltimes \cdots \ltimes S_n$ where each S_i is a *MinMax Recurrent Layer*, defined as a system with dynamics D_i given by a parallel composition of MinMax Recurrent Units, namely $D_i = D_{i,1} \parallel \cdots \parallel D_{i,n_i}$; each $D_{i,j}$ is a *unit* of S_i, and in turn of S; the *state degree* of S is the maximum state dimension among its units $D_{i,j}$. Then S is a *MinMax Recurrent Neural Cascade*, abbreviated as *MinMax RNC*, if (i) each unit $D_{i,j}$ is a MinMax Recurrent Neuron, and (ii) the output function of each layer S_i is an MLP. ∎

4 Expressivity

We characterise the expressivity of MinMax RNCs in terms of the functions over finite alphabets they can implement, using AAT and MAT as presented in §2 and further in §A. We first provide a group-theoretic notion of degree, which we then show to provide a fine-grained characterisation of the expressivity of MinMax RNCs in terms of their state degree.

Definition 3. The *(permutation) degree* of a group G is the minimum number n such that G is isomorphic to a subgroup of the symmetric group S_n. The *degree* of a permutation semiautomaton is the degree of its transition group. The *degree* of a regular function F is the maximum between one and the degree of any simple group that is a factor of a subgroup of the transition monoid of F.

We state the main theorem regarding the expressivity of MinMax RNCs.

Theorem 1. *Every regular function of degree n can be implemented by an η-finite MinMax RNC of state degree n. Every function that is not regular cannot be implemented by a MinMax RNC. Every function that is not group-free cannot be implemented by a MinMax RNC of state degree one.*

Proof sketch. For the first result, implementing a regular function amounts to realising the semiautomaton A of an automaton that implements it. Then A can be realised by a cascade of identity-reset and permutation semiautomata of degree bounded by the degree of the function; in particular, they are all identity-reset in the group-free case. The key step is to show η-finite MinMax Neurons that can realise such semiautomata, as we do in the next section (Theorems 3 and 4). Such Neurons can be composed into a cascade that can realise A and hence implement the given regular function. The second result follows from the fact that, for every MinMax RNC implementing a function over a finite alphabet, there is an equivalent one having a finite state space, implying that the function is regular. For the third result, we observe that, for any fixed scalar values r, s and state x_0, the recurrence $x_t = (r \otimes x_{t-1}) \oplus s$ yields a state sequence that converges to a constant in two steps, meaning that MinMax Units of state degree one cannot realise permutation semiautomata, which prevents them, and cascades thereof, from implementing any function that is not group-free. □

We present a novel, stricter notion of expressivity tha excludes those cases where outputs of a system can only be decoded with a separation margin that vanishes as sequence length increases.

Definition 4. A decoder $\mathrm{dec}: Y \to \Gamma$ *admits a margin* $\gamma \in \mathbb{R}^{>0}$ *on* $Y' \subseteq Y$ if $d_Y(y', y) < \gamma$ implies $\mathrm{dec}(y') = \mathrm{dec}(y)$ for all $y' \in Y'$, $y \in Y$. A system S *can-implement a function* $F : \Sigma^+ \to \Gamma$ *with positive margin* if it can-implement F with some enc and some dec decoder admitting a margin $\gamma \in \mathbb{R}^{>0}$ on the set $Y_S := \{S(\mathrm{enc}(w)) : w \in \Sigma^+\}$; and it *can-recognise a language* $L \subseteq \Sigma^+$ *with positive margin* if it can-implement the characteristic function $\mathbf{1}_L$ with positive margin.

For η-finite systems, positive-margin implementation coincides with the ordinary implementation—see §G.1. Thus, we obtain the following exact characterisation of the expressitivity of MinMax RNCs and of their subclass with state degree one, also under η-finiteness and positive margin requirements.

Theorem 2. *The functions that can be implemented by MinMax RNCs are the regular functions, and the group-free functions for MinMax RNCs of state degree one; the languages that can be recognised by MinMax RNCs are the regular languages, and the star-free languages for MinMax RNCs of state degree one. The two claims hold the same when requiring η-finiteness or a positive margin.*

4.1 Capabilities of MinMax Recurrent Neurons to Realise Fundamental Semiautomata

We show the core abilities underlying the expressivity of MinMax RNCs. First we present Theorem 3, focusing on realising *identity-reset* semiautomata—Example 1 corresponds to this case. Intuitively, it requires storing information and keeping it unchanged as long as needed. More formally, it requires dynamics to have a state transformation $f(\cdot, u)$ acting as an identity, and a set of state transformations $\{f(\cdot, u_i)\}_i$ acting as constant functions. For MinMax Neurons, when $s \leq r$ the transformation $f_{r,s}(x) = (r \odot x) \oplus s$ is identity for all $x \in [s, r]$, and it is constant $f_{r,s}(x) = s$ when $s > r$.

Theorem 3. *Every identity-reset semiautomaton can be realised by an η-finite MinMax Recurrent Neuron having state dimension one.*

Proof sketch. Every identity-reset semiautomaton $A = \langle Q, \Sigma, \delta \rangle$, with states $Q = \{q_i\}_{i=1}^n$ and inputs $\Sigma = \{\sigma_j\}_{j=1}^m$, can be realised by a MinMax Unit $D = \langle X, U, f \mid R, s \rangle$ with states $X = \bigcup_{i=1}^n X_i$ for pairwise-disjoint intervals $X_i = [x_i - \epsilon, x_i + \epsilon]$, scalar inputs $U = \{u_j\}_{j=1}^m$, and input functions

$$R(u_j) := \begin{cases} x_{\mathrm{H}} & \text{if } \delta_{\sigma_j} = \mathrm{id}_Q, \\ x_{\mathrm{L}} & \text{if } \delta_{\sigma_j} = q_\ell, \end{cases} \qquad s(u_j) := \begin{cases} x_{\mathrm{L}} & \text{if } \delta_{\sigma_j} = \mathrm{id}_Q, \\ x_\ell & \text{if } \delta_{\sigma_j} = q_\ell, \end{cases} \qquad \text{for} \quad \begin{array}{l} x_{\mathrm{L}} < x_{\min} := \min X, \\ x_{\mathrm{H}} < x_{\max} := \max X. \end{array}$$

Unit D can realise A as it can represent each state-input pair $\langle q_i, \sigma_j \rangle$ by the pair of η-state and η-input $\langle X_i, u_j \rangle$, over which its state transformation $f(\cdot, u_j)$ is the same as δ_{σ_j}. In fact,

$$f(x, u_j) = \begin{cases} (x_{\mathrm{H}} \otimes x) \oplus x_{\mathrm{L}} = x & \text{if } \delta_{\sigma_j} = \mathrm{id}_Q, \\ (x_{\mathrm{L}} \otimes x) \oplus x_\ell = x_\ell & \text{if } \delta_{\sigma_j} = q_\ell, \end{cases} \quad \rightsquigarrow \quad f(\cdot, u_j) = \begin{cases} \mathrm{id}_X & \text{if } \delta_{\sigma_j} = \mathrm{id}_Q, \\ x_\ell & \text{if } \delta_{\sigma_j} = q_\ell. \end{cases}$$

The same holds as long as the outputs x_{L}, x_{H}, and $\{x_\ell\}_{\ell=1}^n$ of $\langle R, s \rangle$ do not change by more than $\hat{\epsilon} := \min\{\epsilon, x_{\min} - x_{\mathrm{L}}, x_{\mathrm{H}} - x_{\min}\}$, and hence the Unit D can be turned into a MinMax Neuron $\hat{D}$ by replacing $\langle R, s \rangle$ with $\hat{\epsilon}$-approximating MLPs $\langle \hat{R}, \hat{s} \rangle$, that exist since U is compact. □

Next we present Theorem 4, that focuses on realising any permutation semiautomaton A—such as the one in Example 2. It requires dynamics to admit state transformations $\{f(\cdot, u_i)\}_i$ acting as bijections. This could be shown directly using a state dimension equal to the number of states of A—we show it in the appendix. Here we adopt an intermediate characterisation of permutation semiautomata, to show a stronger result connecting the state dimension of a MinMax Neuron to the degree n of A—always bounded by its number of states, but as large as $n!$ in some cases. Semiautomaton A can be realised by a semiautomaton A_{S} where every state is an arrangement $\mathbf{i} = \langle i_1, \ldots, i_n \rangle$ of the indices $[n]$, and each transformation $\delta_\sigma(\mathbf{i})$ rearranges $\mathbf{i}$ into $\langle i_{\pi(1)}, \ldots, i_{\pi(n)} \rangle$ for π a bijection determined by σ. Thus, we can realise A_{S} in place of A, and we can do so by representing directly its n-dimensional states using vectors of $\mathbb{R}^n$. In a MinMax recurrence, the multiplication $x' = R \otimes x$ allows for assigning any coordinate of x to any of x'. For instance, in $x'_\ell = \bigoplus_{k=1}^n (R_{\ell,k} \odot x_k)$, we can obtain $x'_\ell = x_{k^\star}$ by setting $R_{\ell,k^\star}$ to a high value and every other $R_{\ell,k}$ to a low value.

Theorem 4. *Every permutation semiautomaton A can be realised by an η-finite MinMax Recurrent Neuron having state dimension equal to the degree of A.*

Proof sketch. Every semiautomaton A having degree n and inputs $\Sigma = \{\sigma_j\}_{j=1}^m$ is realised by a semiautomaton $A_{\mathrm{S}} = \langle [n]^n, \Sigma, \delta \rangle$ with $\delta(\langle i_1, \ldots, i_n \rangle, \sigma_j) = \langle \pi_j(i_1), \ldots, \pi_j(i_n) \rangle$ where $\pi_j : [n] \to [n]$ is a bijection determined by σ_j. Semiautomaton A_{S}, and hence A, can be realised by the MinMax Unit $D = \langle X^n, U, f \mid R, s \rangle$ having state components $X = \bigcup_{i=1}^n X_i$ for pairwise-disjoint intervals $X_i = [x_i - \epsilon, x_i + \epsilon]$, inputs $U = \{u_j\}_{j=1}^m$, and input functions given component-wise by

$$R(u_j)_{\ell,k} := \begin{cases} x_{\mathrm{H}} & \text{if } \pi_j(\ell) = k, \\ x_{\mathrm{L}} & \text{otherwise}, \end{cases} \qquad s(u_j)_\ell := x_{\mathrm{L}}\,, \qquad \text{for} \quad \begin{array}{l} x_{\mathrm{L}} < x_{\min} := \min X, \\ x_{\mathrm{H}} < x_{\max} := \max X. \end{array}$$

Unit D can realise A as it can represent each state-input pair $\langle \mathbf{i}, \sigma_j \rangle$ with $\mathbf{i} = \langle i_1, \ldots, i_n \rangle$ by the η-pair $\langle X_{\mathbf{i}}, u_j \rangle$ with $X_{\mathbf{i}} = X_{i_1} \times \cdots \times X_{i_n}$, over which its state transformation $f(\cdot, u_j)$ permutes coordinates in the same way as δ_{σ_j}. In fact, each transformation $f(\cdot, u_j)$ amounts to the linear operator given by the permutation matrix P_{π_j} representing π_j, with entries $(P_{\pi_j})_{\ell,k} = \mathbf{1}[\pi_j(\ell) = k]$.

$$f(x, u_j)_\ell = \big(x_{\mathrm{H}} \odot x_{\pi_j(\ell)}\big) \oplus \big(\bigoplus\nolimits_{k \neq \pi_j(\ell)} x_{\mathrm{L}} \odot x_k\big) \oplus x_{\mathrm{L}} = x_{\pi_j(\ell)} \quad \rightsquigarrow \quad f(\cdot, u_j) = P_{\pi_j} \cdot x \quad (1)$$

The same holds as long as the output components x_{L} and x_{H} of $\langle R, s \rangle$ do not change by more than $\hat{\epsilon} := \min\{\epsilon, x_{\min} - x_{\mathrm{L}}, x_{\mathrm{H}} - x_{\min}\}$, and hence the Unit D can be turned into a MinMax Neuron $\hat{D}$ by replacing $\langle R, s \rangle$ with $\hat{\epsilon}$-approximating MLPs $\langle \hat{R}, \hat{s} \rangle$, that exist since U is compact. □

5 Complexity

We analyse the sequential and parallel complexity of evaluating MinMax RNCs. First we analyse the complexity of evaluating MinMax recurrence for T steps, which amounts to computing all states $x_t = (R_t \otimes x_{t-1}) \oplus s_t$ given $x_0 \in \mathbb{R}^{d_s}$, $R_t \in \mathbb{R}^{d_s \times d_s}$, and $s_t \in \mathbb{R}^{d_s}$ for all $t \in [1..T]$.

Theorem 5. *MinMax recurrence can be evaluated sequentially in time $O(T \cdot d_s^2)$, and also in parallel with depth $O(\log T \cdot \log d_s)$ and work $O(T \cdot d_s^3)$.*

Proof sketch. The sequential algorithm that computes states by evaluating $x_t = (R_t \otimes x_{t-1}) \oplus s_t$ for t from 1 to T runs in the specified time since the most expensive operation at each step is the matrix-vector multiplication $R_t \otimes x_{t-1}$ which can be done in $O(d_s^2)$. To obtain a parallel algorithm, we observe that letting $\tau_t(x) := (R_t \otimes x) \oplus s_t$, we have $x_t = (\tau_t \circ \cdots \circ \tau_1)(x_0)$, and each composition $\tau_t \circ \cdots \circ \tau_1$ can be computed via *parallel scan*, cf. [31, 39, 40]. Specifically, parallel scan evaluates all prefixes of an expression over any given monoid—i.e., expressions where several elements are combined via an associative operator, for which some element acts as identity. In our case, the relevant monoid is the finite set of matrix-vector pairs $\langle R, s\rangle$ over the scalars of x_0, R_t, s_t, equipped with the associative operator $\langle R, s\rangle \circledast \langle R', s'\rangle := \langle R' \otimes R,\ (R' \otimes s) \oplus s'\rangle$. Then we have $(\tau_t \circ \cdots \circ \tau_1)(x) = (R_{1:t} \otimes x) \oplus s_{1:t}$ with $\langle R_{1:t}, s_{1:t}\rangle = \langle R_1, s_1\rangle \circledast \cdots \circledast \langle R_t, s_t\rangle$ that can be computed via parallel scan. The complexity bound holds since parallel scan performs $O(\log T)$ rounds of depth $O(\log d_s)$, and it carries out $O(T)$ compositions each of work $O(d_s^3)$. □

The above results for MinMax recurrence make it straightforward to derive algorithms and bounds for of *evaluating MinMax RNCs*, that is to compute the output sequence of a MinMax RNC S over an input u. We provide bounds in terms of the input-sequence length T, number n_L of layers, state degree d_s , and maximum dimension d across the input space, output space, hidden layers of its MLPs, and state space of any layer. The bounds consider the number of hidden MLP layers as fixed, in order to be more intelligible—general bounds in §K.

Theorem 6. *MinMax RNCs can be evaluated sequentially in time $O\big(T \cdot n_L \cdot d \cdot (d + d_s^2)\big)$, and also in parallel with work $O\big(T \cdot n_L \cdot d \cdot (d + d_s^2)\big)$ and depth $O\big(n_L \cdot (\log T \cdot \log d_s + \log d)\big)$.*

The following bound highlights the dependency on the sequence length T, obtained immediately from the above since work and depth translate to runtime via Brent's Scheduling Principle, cf. [41].

Corollary 1. *MinMax RNCs of fixed size can be evaluated in time $O(\log T)$ over $\Theta(T)$ processors.*

6 Stability and Gradients

We study stability and gradient properties of parametric MinMax RNCs where parameters are the ones of their MLPs—see §H.2. Parametric MinMax RNCs have states and outputs that are bounded uniformly over bounded input and parameter values, independently of input-sequence length.

Theorem 7. *Let S be a parametric MinMax RNC with U and Θ its input and parameter spaces. For every $r \in \mathbb{R}^{\geq 0}$, there is $B_r \in \mathbb{R}$ such that, for every $\theta \in \Theta_{\leq r}$, on every input sequence $(u_t)_{t=1}^T \in U^+_{\leq r}$, the instance S_θ has states $(x_t)_{t=1}^T$ and outputs $(y_t)_{t=1}^T$ satisfying $\|x_t\|, \|y_t\| \leq B_r$ for all $t \in [T]$.*

Proof sketch. The key observation is that each MinMax recurrence step $x_t = (R_t \otimes x_{t-1}) \oplus s_t$ satisfies the one-step max-norm bound $\|x_t\|_{\max} \leq \max\{\|R_t\|_{\max}, \|s_t\|_{\max}\}$, which is independent of $\|x_{t-1}\|_{\max}$. The reset and set values are produced by MLPs, which are bounded on bounded inputs. Boundedness in Euclidean or max norm implies boundedness in the other. Thus, we obtain a bound on the states x_t that is uniform in t by applying the above one-step bound to each recurrent unit and proceeding inductively through the layers of the RNC, with base case given by the input sequence $u = (u_t)_{t=1}^T$ satisfying $\max_t \|u_t\| \leq r$, Then, a bound on the outputs y_t follows from boundedness of the MLPs on the resulting bounded states, inputs, and parameters. □

To study gradient properties, we note that ordinary derivatives may not exist everywhere due to the min and max operations in the recurrence. Such operations consist of *MinMax reductions* $\min_n / \max_n$ that map scalars $x_1, \ldots, x_n$ to the corresponding minimum/maximum value $x_\star$. We extend the ordinary derivative operator to the *pseudo-derivative operator* $\tilde{\mathsf{D}}_\circledast$, that assigns $\frac{1}{k}$ to each the k *active*

variables $x_i = x_\star$ of a MinMax reduction and zero to the others—thus mapping $\min_n / \max_n$ to its *minimum-norm sub/supergradient*, cf. [42]. Then, the pseudo-derivative operator $\tilde{\mathsf{D}}_\circledast$ is extended to the other relevant functions via the *chain rule*. However, different chain-rule derivations may yield different pseudo-derivatives. We formalise this dependency by parametrising the operator $\tilde{\mathsf{D}}_\circledast$ with a choice of a *selector* Γ over *computation graphs*, obtaining what we call a *selected pseudo-derivative operator* $\tilde{\mathsf{D}}^\Gamma_\circledast$, that applies the chain rule according to the graph selected by Γ, and applies the operator $\tilde{\mathsf{D}}_\circledast$ locally at the level of nodes. Computation graphs are taken from a class $\mathcal{G}_\circledast$ we identify, that includes the graphs arising from a broad variety of evaluation strategies, and in particular from sequential and parallel-scan evaluation—thus covering the algorithms underlying our complexity results. Notably, our operator $\tilde{\mathsf{D}}^\Gamma_\circledast$ is in line with of *Pytorch's AutoDiff* [34] when applied to MinMax RNCs, and hence we may expect our results to provide a reasonably-faithful characterisation of the empirical behaviour of MinMax RNCs implemented in PyTorch.

We show next that the pseudo-derivatives of loss functions of MinMax RNCs are bounded uniformly on bounded inputs and parameters, and they coincide with the ordinary derivative almost everywhere. For S a parametric MinMax RNC with U and Y its input and output spaces, its *sample space* $\mathcal{Z} := U^+ \times Y$ its *r-bounded sample space* $\mathcal{Z}_{\leq r} := U^+_{\leq r} \times \hat{Y}_{\leq r}$, and the *loss function* $\mathcal{L}^\ell_z(\theta) := \ell(\check{S}(u;\theta), y)$ for a given sample $z = (u, y) \in \mathcal{Z}$, under a choice of $\ell : Y^2 \to \mathbb{R}$.

Theorem 8 (Boundedness and correctness of loss pseudo-gradients)**.** *Let S be a MinMax RNC with parameter space Θ, let $\ell : Y^2 \to \mathbb{R}$ be analytic, and let Γ be a graph selector over $\mathcal{G}_\circledast$.* (Boundedness): *for every $r \in \mathbb{R}^{\geq 0}$, there is $B_r \in \mathbb{R}$ such that $\|\tilde{\mathsf{D}}^\Gamma_\circledast \mathcal{L}^\ell_z(\theta)\| \leq B_r$ for all $z \in \mathcal{Z}_{\leq r}$, $\theta \in \Theta_{\leq r}$.* (Correctness): *for every $z \in \mathcal{Z}$, the equality $\tilde{\mathsf{D}}^\Gamma_\circledast \mathcal{L}^\ell_z(\theta) = \mathsf{D}\mathcal{L}^\ell_z(\theta)$ holds for almost every $\theta \in \Theta$.*

Proof sketch. (Boundedness): the key observation is that MinMax recurrence propagates states exclusively via min/max reductions, whose sub/supergradients have non-negative entries summing to one, and hence amount to non-expansive linear operators that do not contribute in increasing derivative norms. Thus, derivative norms do not increase across time and can only increase across layers, where growth is determined by derivatives of MLPs, that are bounded on bounded inputs as they are continuously-differentiable, and boundedness of inputs holds by Theorem 7. *(Correctness):* the pseudo-derivative $\tilde{\mathsf{D}}^\Gamma_\circledast \mathcal{L}^\ell_z$ is an *intensional derivative* [43], guaranteed to coincide with the ordinary derivative $\mathsf{D}\mathcal{L}^\ell_z$ almost everyhere. It is an intensional derivative since so are minimum-norm sub/supergradients of min/max reductions and the ordinary derivatives of MLPs, and consequently the pseudo-derivatives obtained by $\tilde{\mathsf{D}}^\Gamma_\circledast$ via chain-rule derivations. □

We next show that pseudo-derivatives of MinMax RNCs need not vanish over arbitrarily long horizons, while coinciding with ordinary derivatives. Notably, this holds for the MinMax Neurons underlying our expressivity results, since they have exact identity and permutation state transformations, which correspond to linear isometries.

Theorem 9 (Non-vanishing state-to-state derivatives)**.** *For every $\epsilon \in \mathbb{R}^{>0}$, there exists $(\mathcal{D}_n)_{n \geq 1}$ satisfying the following jointly: (i) each $D \in \mathcal{D}_n$ is a MinMax Neuron whose state space contains an n-dimension hypercube $\tilde{X}_D$ of side length ϵ; (ii) every identity-reset semiautomaton can be realised by a Neuron of $\mathcal{D}_1$; (iii) every n-degree permutation semiautomaton can be realised by a Neuron of $\mathcal{D}_n$; (iv) each $D \in \mathcal{D}_n$ admits an infinite input squence u such that, for every selector Γ over $\mathcal{G}_\circledast$,*

$$\tilde{\mathsf{D}}^\Gamma_\circledast \check{D}^u_t(x_0) = \mathsf{D}\check{D}^u_t(x_0) \quad \textit{and} \quad \|\tilde{\mathsf{D}}^\Gamma_\circledast \check{D}^u_t(x_0)\| = 1 \qquad \forall x_0 \in \tilde{X}_D \quad \forall t \in \mathbb{N}^{>0}.$$

Proof sketch. Neurons $D \in \mathcal{D}_1$ that can realise identity-reset semiautomata are from Theorem 3. We choose $u = u_j \cdots u_j$ for u_j inducing an identity state transformation $f(\cdot, u_j)$, and so have that $\check{D}^u_t$ is identity for every t, with derivative $\mathsf{D}\check{D}^u_t(x_0) = 1$. Neurons $D \in \mathcal{D}_n$ that can realise n-degree permutation semiautomata are from Theorem 4. We choose $\tilde{X}_D = X_1 \times \cdots \times X_n$ with X_i pairwise-disjoint intervals, and so have that every $f(\cdot, u_t)$ is a coordinate permutation represented by a permutation matrix P_t as per (1). Then also $\mathsf{D}\check{D}^u_t$ is a coordinate permutation, represented by the permutation matrix $P_t \cdots P_1$, that has unit norm and gives to the derivative $\mathsf{D}\check{D}^u_t = P_t \cdots P_1$. In both cases, the derivative coincides with $\tilde{\mathsf{D}}^\Gamma_\circledast \check{D}^u_t(x_0)$ since the constructions ensure that no ties occur among the arguments of min/max operators and, in the permutation case, each coordinate of $\check{D}^u_t(x_0)$ is a unique coordinate of x_0 by the choice of $\tilde{X}_D$. □

7 Empirical Evaluation

We evaluate MinMax RNCs [44] implemented in PyTorch [45] run on a single A100 GPU—see §O, P.

7.1 Evaluation on group-free tasks

Our first goal is to assess the capabilities of MinMax RNCs of degree one to learn group-free tasks.

Benchmarks. We consider three families of group-free tasks that test fundamental capabilities with a focus on long horizons. `Latching`(n) is based on [46] and consists in returning the first element of the input sequence, generated from a set of $20n$ tokens, with n that can appear first. It focuses on storing and retaining information over long horizons, with n increasing the amount of information. `Sequences`(n) is novel and consists in simultaneously matching two patterns E_1 and E_2 consisting of n events to be matched in sequence, with the j-th event of E_i being a token from a given set $E_{i,j}$, and restarting to match E_i when encountering a token from a given set I_i. It focuses on the ability to solve inherently sequential tasks, requiring to combine n pieces of information scattered across the input sequence, and hence to maintain intermediate information over long horizons. `InductionHeads`(n) is from [13, 47] and consists in recalling, at the second occurrence of a designated marker, the token that has appeared right after the first occurrence of the marker; with n possible tokens to recall, the marker's first occurrence within the first 30 steps for training, and 50 for validation and evaluation. It focuses on the ability to store, retain, and recall information over long horizons, with n increasing the amount of information.

Baselines. We include a representative selection of SOTA baselines covering different forms of recurrence: Mamba [14], mLSTM [6], xLSTM-L [32], based on contractive linear recurrence; DeltaNet [24] based on non-expansive linear recurrence; sLSTM [6] based on nonlinear recurrence.

Setup. We train on sequences of length up to 512, validate up to 2048, and evaluate on significantly longer sequences up to length 1M; with the rationale that the ability of a model to generalise from length 512 to 1M should reasonably indicate whether it has learned to solve the task in general, or only its restriction to sequences of a bounded length. The training configuration for MinMax is: Adam optimiser with weight decay $1\mathrm{e}{-4}$, constant learning rate $1\mathrm{e}{-3}$, batch size 64. MinMax uses AMP and PyTorch's compile. The baselines are trained according to the respective papers. In the case of multiple hyperparameter configurations, we evaluate the model having the highest validation accuracy. The time limit is 4h for each model-benchmark pair. Datasets have size 20k for training, 1k for validation, and 1k for evaluation. Models are set to comparable architectural hyperparameters: fixed for `Latching`(n) and `InductionHeads`(n) with 2 layers and ~125k params; the same for `Sequences`(n) but with $n+1$ layers—all models fail to learn with fewer layers. `Latching`(n) and `Sequences`(n) are seen as seq-to-seq tasks, averaging loss and accuracy across sequence steps. For `InductionHeads`(n) we measure loss and accuracy at the recall step.

Results. Table 1 summarises the results on group-free tasks. MinMax always achieves perfect validation accuracy and retains it at evaluation time, whereas the other models can achieve perfect or near-perfect validation accuracy but fail to retain it at evaluation time as it decreases over sequence length, as shown in Figure 2; this validates the formal-expressivity results for MinMax RNCs of

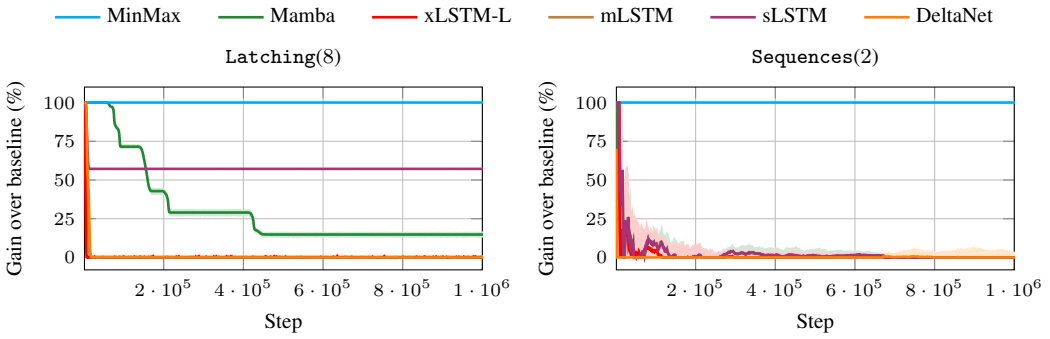


Figure 2: Accuracy gain $\max\left(0, \frac{\mathrm{acc}-\mathrm{acc}_{\mathrm{naive}}}{1-\mathrm{acc}_{\mathrm{naive}}}\right)$ over a naive baseline that ignores inputs and predicts the most likely label at each step: 100% is perfect accuracy, and 0% is no better than the baseline.

Table 1: *Evaluation accuracy* (%): means of 99% CIs with max margin 1.7. *Relative throughput* (tok/s, ratio w.r.t. MinMax): avg. of training and evaluation ratios of the means of 99% CIs with max margin 0.4. For `Latching`(2^n) and `Sequences`(m), avg. over $n \in \{2, \dots, 9\}$ and $m \in \{2, 3, 4, 8, 16\}$.

	MinMax		Mamba		xLSTM-L		mLSTM		sLSTM		DeltaNet	
	Acc	Thr	Acc	Thr	Acc	Thr	Acc	Thr	Acc	Thr	Acc	Thr
`Latching`(2^n)	100	1	54	0.3	8	1.1	7	0.1	28‡	0	8	0.6
`Sequences`(m)	100	1	47	0.9	47	1.9	45	0.2	47‡	0	17	1.2
`InductionHeads`(16)	100	1	100§	0.3	7†	1.2	7	0.1	6	0	5	1.0
`InductionHeads`(30)	100	1	4†	0.3	3†	1.2	3	0.1	3	0	4	1.3

† no training seed succeeded (val. acc. <50%) ‡ training timed out § training succeeded only with batch size 8

state degree one (Theorem 2), and indirectly the results on their favourable stability and gradient properties (Theorems 7–9). MinMax has the second-highest training throughput after xLSTM-L, and the highest throughput at evaluation time. This validates the computational-complexity results for MinMax RNCs of state-degree one (Theorem 5). We note that the throughput of MinMax is significantly lower when not using AMP and PyTorch's compile, which suggests that MinMax recurrence is effectively able to take advantage of hardware acceleration in a significant manner.

Assessment of non-vanishing gradient. Successful training over `Sequences`(n) requires state gradients not to vanish across steps between the event tokens to match. Calling *range dependency* the mean number of such steps in a training sample, the above experiments on `Sequences`(2) have a range dependency of 376 steps. We increase it to 120k steps and observe that MinMax still achieves near-perfect validation and evaluation accuracies, whereas the validation accuracies of Mamba and DeltaNet drop from perfect to ~94%. This suggests that the higher accuracy of MinMax may be attributed to its state gradients being more effective in propagating error signals across long range dependencies, hence providing empirical support to the non-vanishing gradient results of Theorem 9.

7.2 Evaluation on group-based tasks

To evaluate MinMax RNCs beyond group-free tasks, we consider the *word problems* defined by the symmetry groups S_3 and S_4, included in the evaluation of DeltaProduct by Siems et al. [25], which thus provides us with a baseline. We set MinMax to have a single layer and state degree equal to 3 and 4 on S_3 and S_4, respectively, which suffices according to Theorem 1. We train on 200k sequences of length 128, validate on 1k sequences of length 256, and evaluate on 1k sequences of length 1M. MinMax achieves perfect validation and evaluation accuracies. In contrast, the single-layer DeltaProduct results reported in Fig. 5 of [25] show an accuracy that begins to deteriorate already at length 400–500 on S_3 and length 200–400 on S_4, despite training over datasets of 2M examples.

7.3 Next-token prediction on OpenWebText (OWT)

We train a degree-one MinMax RNC of 112M parameters for next-token prediction following [48], which reproduces GPT-2 on OWT [49]. It achieves the same validation loss of 3.12 as the one reported in [48] for the released OpenAI GPT-2 124M checkpoint when evaluated on OWT.

8 Conclusion

The theoretical analysis and empirical evaluation suggest that MinMax recurrence is a relevant alternative to the existing forms of recurrence. Our future work will focus on better understanding the effectiveness of MinMax RNCs in real-world applications, and also on the potential of MinMax recurrence as a mechanism to integrate existing architectures in order to improve their capabilities.

Limitations. We acknowledge that (i) our empirical evaluation is relatively limited compared to the ones included in related research papers; (ii) we do not provide precise answers regarding the depth and width required for cascades to recognise a given language, due to the corresponding questions being open in automata theory.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[2] David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer encoders. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5544–5562. PMLR, 2023. URL https://proceedings.mlr.press/v202/chiang23a.html.

[3] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 0364-0213. doi: 10.1016/0364-0213(90)90002-E. URL https://www.sciencedirect.com/science/article/pii/036402139090002E.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, 1997. doi: 10.1162/NECO.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

[6] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

[7] Nadezda Alexandrovna Knorozova and Alessandro Ronca. On the expressivity of recurrent neural cascades. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 10589–10596. AAAI Press, 2024. doi: 10.1609/AAAI.V38I9.28929. URL https://doi.org/10.1609/aaai.v38i9.28929.

[8] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/pascanu13.html.

[9] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=8GjSf9Rh7Z.

[10] Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. Decimamba: Exploring the length extrapolation potential of mamba. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iWSl5Zyjjw.

[11] Igor Dubinin, Antonio Orvieto, and Felix Effenberger. Improved state mixing in higher-order and block diagonal linear recurrent networks, 2026. URL https://openreview.net/forum?id=fF72TYOKjZ.

[12] Ting-Han Fan, Ta-Chung Chi, and Alexander Rudnicky. Advancing regular language reasoning in linear recurrent neural networks. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 45–53, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-short.4. URL https://aclanthology.org/2024.naacl-short.4/.

[13] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

[14] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/102f0bb6efb3a6128a3c750dd16729be-Abstract.html.

[15] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

[16] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

[17] Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "goose" with expressive dynamic state evolution. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayB1PACN5j.

[18] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936. URL https://aclanthology.org/2023.findings-emnlp.936/.

[19] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

[20] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 26670–26698. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/orvieto23a.html.

[21] Antonio Orvieto, Soham De, Caglar Gulcehre, Razvan Pascanu, and Samuel L Smith. Universality of linear recurrences followed by non-linear projections: Finite-width guarantees and benefits of complex eigenvalues. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller,

Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 38837–38863. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/orvieto24a.html.

[22] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 28043–28078. PMLR, 2023. URL https://proceedings.mlr.press/v202/poli23a.html.

[23] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=y6SqbJfCSk.

[24] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 9355–9366. PMLR, 2021. URL http://proceedings.mlr.press/v139/schlag21a.html.

[25] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear RNNs via householder products. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=SoRiaijTGr.

[26] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=wXfuOj9C7L.

[27] Benjamin Walker, Lingyi Yang, Nicola Muca Cirone, Cristopher Salvi, and Terry Lyons. Structured linear CDEs: Maximally expressive and parallel-in-time sequence models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=HKDyRDzy1E.

[28] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

[29] Annan Yu and N. Benjamin Erichson. Block-biased mamba for long-range sequence processing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=5WKEH9LhAQ.

[30] Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=jY4PhQibmg.

[31] Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, Nov 1990. URL https://www.cs.cmu.edu/afs/cs.cmu.edu/project/scandal/public/papers/CMU-CS-90-190.html. Accessed 21 Oct 2026.

[32] Maximilian Beck, Korbinian Pöppel, Phillip Lippe, Richard Kurle, Patrick M Blies, Günter Klambauer, Sebastian Böck, and Sepp Hochreiter. xLSTM 7b: A recurrent LLM for fast and efficient inference. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LV3DpKDO8B.

[33] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=y8Rm4VNRPH.

[34] PyTorch. Autograd mechanics. https://docs.pytorch.org/docs/stable/notes/autograd.html, 2026. Accessed: 2026-04-18.

[35] Michael A. Arbib. *Theories of abstract automata*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, N.J, 1969.

[36] Juris Hartmanis and R. E. Stearns. *Algebraic structure theory of sequential machines*. Prentice-Hall international series in applied mathematics. Prentice-Hall, Englewood Cliffs, N.J, 1966.

[37] Adam Dankowiakowski and Alessandro Ronca. Metric automata theory: A unifying theory of RNNs. In *NeurIPS*, 2025.

[38] Yoshua Bengio, Ian Goodfellow, Aaron Courville, et al. volume 1. MIT Press, 2017.

[39] G.E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11): 1526–1538, 1989. doi: 10.1109/12.42122.

[40] Peter M Kogge and Harold S Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE transactions on computers*, 100(8):786–793, 2009.

[41] Guy E. Blelloch and Bruce M. Maggs. Parallel algorithms. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and Theory of Computation Handbook*, chapter 25. Chapman & Hall/CRC, 2010.

[42] R Tyrrell Rockafellar. *Convex analysis*, volume 28. Princeton university press, 1997.

[43] Wonyeol Lee, Hangyeol Yu, Xavier Rival, and Hongseok Yang. On correctness of automatic differentiation for non-differentiable functions. *Advances in neural information processing systems*, 33:6719–6730, 2020.

[44] MinMax recurrent neural cascades, 2026. URL https://github.com/minmaxrnc/model. GitHub repository. Accessed: 2026-05-28.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc., 2019. URL https://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.

[46] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[47] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[48] Andrej Karpathy. nanoGPT, 2022. URL https://github.com/karpathy/nanoGPT. GitHub repository. Accessed: 2026-05-04.

[49] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[50] Steven G Krantz and Harold R Parks. *A primer of real analytic functions*. Springer Science & Business Media, 2002.

[51] Brian A Davey and Hilary A Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002. doi: 10.1017/CBO9780511809088.

[52] S. Burris and H.P. Sankappanavar. *A Course in Universal Algebra*. Graduate Texts in Mathematics. Springer New York, 2011. ISBN 9781461381327.

[53] John M Howie. *Fundamentals of Semigroup Theory*. Oxford University Press, 12 1995. ISBN 9780198511946. doi: 10.1093/oso/9780198511946.001.0001. URL https://doi.org/10.1093/oso/9780198511946.001.0001.

[54] John D. Dixon and Brian Mortimer. The primitive permutation groups of degree less than 1000. *Mathematical Proceedings of the Cambridge Philosophical Society*, 103(2):213–238, 1988. doi: 10.1017/S0305004100064793.

[55] Abraham Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968.

[56] Pál Dömösi and Chrystopher L Nehaniv. *Algebraic theory of automata networks: An introduction*. SIAM, 2005.

[57] Stephen Willard. *General Topology*. Dover Publications, 2004. ISBN 978-0-486-43479-7.

[58] Gerald B. Folland. *Real Analysis: Modern Techniques and Their Applications*. John Wiley & Sons, New York, 2nd edition, 1999. ISBN 978-0-471-31716-6.

[59] Markus Holzer and Sebastian Jakobi. More structural characterizations of some subregular language families by biautomata. In Zoltán Ésik and Zoltán Fülöp, editors, *Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014*, EPTCS, pages 271–285, 2014. doi: 10.4204/EPTCS.151.19. URL https://doi.org/10.4204/EPTCS.151.19.

[60] Geonho Hwang, Yeachan Park, Wonyeol Lee, and Sejun Park. Floating-point neural networks can represent almost all floating-point functions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 26412–26432. PMLR, 13–19 Jul 2025.

[61] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and Real Computation*. Springer, 1998. doi: 10.1007/978-1-4612-0701-6.

[62] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[63] Alessandro Ronca. MinMaxRNC–v0.3.0, 2026. URL https://pypi.org/project/minmaxrnc/0.3.0/. Python package. Accessed: 2026-05-28.

[64] Alessandro Ronca. Minmax recurrent neural cascades, 2026. URL https://arxiv.org/abs/2605.06384.

[65] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Workshop on Spoken Language Translation*, 2019.

[66] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

[67] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 933–941. PMLR, 2017.

[68] Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

[69] Albert Gu and Tri Dao. Mamba, 2024. URL https://pypi.org/project/mamba-ssm/2.2.4/. Python package. Accessed: 2026-05-26.

[70] Maximilian Beck. xLSTM, 2025. URL https://pypi.org/project/xlstm. Python package. Accessed: 2026-05-26.

[71] Alessandro Ronca. MinMaxRNC–v0.1.0, 2026. URL https://pypi.org/project/minmaxrnc/0.1.0/. Python package. Accessed: 2026-05-28.

[72] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. GatedDeltaNet: Official pytorch implementation of gated delta networks: Improving mamba2 with delta rule, 2024. URL https://github.com/NVlabs/GatedDeltaNet. Initial code release 9 December 2024; no tagged release; main branch (unversioned). Accessed 23 Sep 2026.

[73] Songlin Yang and Yu Zhang. FLA: A triton-based library for hardware-efficient implementations of linear attention mechanism, 2024. URL https://github.com/fla-org/flash-linear-attention. Version 0.5.2; commit 9c8e42e; date 2026-07-27. Accessed 23 Sep 2026.

[74] FLA Team. Flash linear attention, 2024. URL https://github.com/fla-org/flash-linear-attention. GitHub repository. Accessed: 2026-08-31.

[75] Endre Süli and David F Mayers. *An introduction to numerical analysis*, volume 1. Cambridge university press Cambridge, 2003.

Table of Contents

I Background 22

A Extended Background 23

A.1 Notation . . . 23
A.2 Functions . . . 23
A.3 Sequences . . . 26
A.4 Vectors and Matrices . . . 28
A.5 Notation for Derivatives . . . 29
A.6 Partitions of a Set . . . 32
A.7 Affine Functions . . . 33
A.8 Piecewise Representations of Functions . . . 33
A.9 Piecewise-Affine Functions . . . 34
A.10 Analytic Euclidean Maps and Neural Primitives . . . 37
A.11 Affine Hyperplanes and Semilinear Sets . . . 40
A.12 PAP Functions and Intensional Derivatives . . . 42
A.13 Lattice Theory . . . 44
A.14 Algebraic Structures . . . 45
A.15 Semigroup and Group Theory . . . 47
A.16 Algebraic Automata Theory . . . 49
A.17 Metric Spaces and Ambient Spaces . . . 50
A.18 Measure Theory . . . 52
A.19 Metric Automata Theory . . . 54
A.20 Definite Languages . . . 55
A.21 Graphs, Isomorphisms, and Renaming of Nodes . . . 56

II Preliminary Notions and Results 58

B Additional on Dynamics and Systems 59

B.1 Parametric Dynamical Systems . . . 59
B.2 Composition Operators for Dynamics and Systems . . . 60
B.3 Composition Operators for Parametric Dynamics and Systems . . . 61
B.4 Properties of Composition Operators for Parametric Dynamics and Systems . . . 65
B.5 Euclidean Dynamics and Systems . . . 69
B.6 State Functions of Dynamics and Systems . . . 70

C Generic Architectures 71

C.1 Recurrent Layers . . . 71
C.2 Recurrent Cascades . . . 72
C.3 Recurrent Neural Cascades (RNCs) . . . 73
C.4 Remarks on Convolution . . . 76

D MinMax Algebra for MinMax Recurrences **80**
D.1 Scalar MinMax Algebra . . . 80
D.2 Componentwise MinMax Operations . . . 81
D.3 MinMax Matrix Algebra . . . 82
D.4 MinMax Affine Operators . . . 83
D.5 MinMax Recurrences . . . 84
D.6 Finite Restrictions of the MinMax Algebra . . . 85

E Selected Intensional-Derivative Operators **89**
E.1 Computation Graphs . . . 89
E.2 Pseudo-derivative Operators . . . 100
E.3 Selected Pseudo-Derivative Operators . . . 104
E.4 Intensional-Derivative Operators . . . 105
E.5 Graph-Induced Intensional-Derivative Operators . . . 105
E.6 Selected Intensional-Derivative Operators . . . 108
E.7 Analytic Extensions of Computation Graphs . . . 110

F Properties: Expressivity, BIBS, BIBO, BIBSD, BIBLG, and nnVSD **112**
F.1 Expressivity . . . 112
F.2 BIBS and BIBO . . . 114
F.3 BIBSD: Bounded-Input Bounded-State-Derivative . . . 115
F.4 BIBLG: Bounded-Input Bounded-Loss-Gradient . . . 117
F.5 nnVSD: Non-necessarily Vanishing State Derivative . . . 119

G Architecture-agnostic Results: Expressivity, BIBS, BIBO, BIBSD, BIBLG **120**
G.1 Results for Expressivty . . . 120
G.2 Results for BIBS and BIBO . . . 122
G.3 Results for BIBSD and BIBLG . . . 126

H Additional Notions for MinMax RNCs **133**
H.1 MinMax Recurrence Operator . . . 133
H.2 Parametric MinMax RNCs . . . 133
H.3 Floating-point MinMax RNCs . . . 134

III Theoretical Results for MinMax RNCs 136

I Expressivity of MinMax RNCs **137**
I.1 Preliminary Results . . . 137
I.2 Realising semiautomata using single MinMax Recurrent Units . . . 144
I.3 Realising Cascades . . . 149
I.4 Formal Expressivity (Proof of Theorem 1) . . . 151
I.5 Positive-margin expressivity of MinMax RNCs . . . 156

J Algorithms for MinMax RNCs **157**

J.1 Preliminary remarks on our Pseudocode . . . 157
J.2 Auxiliary Algorithms . . . 157
J.3 Sequential evaluation . . . 157
J.4 Parallel Evaluation . . . 158

K Computational Complexity of MinMax RNCs **165**
K.1 Model of computation . . . 165
K.2 Computation problems . . . 165
K.3 Proofs . . . 167

L BIBS and BIBO Stability of MinMax RNCs **189**
L.1 A one-step MinMax state-value bound . . . 189
L.2 Uniform-BIBS of MinMax recurrent units and of MinMax RNCs . . . 189
L.3 BIBS and BIBO of MinMax RNCs . . . 190

M Gradients of MinMax RNCs **192**
M.1 Preliminaries . . . 192
M.2 Relevant Class of Functions . . . 192
M.3 Pseudo-differentiating Min and Max Functions . . . 193
M.4 Analytic-MinMax Computation Graphs . . . 197
M.5 Virtual Graph-Extension Notation for Analytic-Minax Graphs . . . 199
M.6 Properties of the Analytic-MinMax Pseudo-Derivative Operator . . . 202
M.7 MinMax Recurrence Computation Graphs . . . 204
M.8 MinMaxRNC State Computation Graphs . . . 206
M.9 Sequential Computation Graphs for MinMax RNCs . . . 210
M.10 Parallel-Scan Computation Graphs for MinMax RNCs . . . 212
M.11 BIBSD: Uniform Selected uBIBSD for MinMax RNCs . . . 219
M.12 BIBLG: Bounded Loss Gradients of MinMax RNCs . . . 222
M.13 Non-necessarily Vanishing State-to-State Derivatives . . . 227
M.14 Deferred Proofs for MinMax Pseudo-Derivatives . . . 231

N Piecewise-Affine Structure of MinMax Recurrences **237**
N.1 Active Variables of MinMax Functions . . . 238
N.2 Piecewise Propagate-Reset Structure of MinMax Recurrences . . . 240
N.3 Composition and Relation to Linear Recurrences . . . 247

IV Implementation and Experiments 251

O Implementation of MinMax RNCs **252**
O.1 MinMax Neuron . . . 252
O.2 MinMax Layer . . . 253
O.3 MinMax RNC . . . 253
O.4 Language Model Wrapper . . . 253

O.5 Initialisation . . . 253
O.6 Feedforward Networks . . . 254

P Details of the Experimental Evaluation on Group-free Benchmarks 256
P.1 Benchmarks . . . 256
P.2 Models . . . 259
P.3 Trainers . . . 261
P.4 Training Protocol . . . 262
P.5 Evaluation Protocol . . . 263
P.6 Overview of the Included Experiments . . . 263
P.7 Results: Accuracy . . . 263
P.8 Results: Throughput . . . 267

Q Details of Next-Token Prediction on OpenWebText with MinMax RNCs 268
Q.1 Setup . . . 268
Q.2 Results . . . 268

V Technical Analysis of Related Architectures 271

R Analysis of Linear RNNs 272
R.1 Linear RNNs . . . 272
R.2 General Linear Recurrent Units . . . 273
R.3 Contractive Linear RNNs . . . 284
R.4 Smoothness up to the Closure . . . 286
R.5 Positive-Margin Expressivity of Smooth Contractive Linear RNNs . . . 291
R.6 BIBS of Smooth Contractive Linear RNNs . . . 293
R.7 BIBSD and nnVSD of Smooth Contractive Linear RNNs . . . 293

S Analysis of DeltaNet 296
S.1 Definition of DeltaNet . . . 296
S.2 Overview of the Analysis . . . 298
S.3 BIBS Instability of DeltaNet . . . 298
S.4 BIBSD Instability of DeltaNet . . . 300
S.5 BIBLG Instability of DeltaNet . . . 303
S.6 DeltaNet: Technical Lemmas on Instability . . . 307
S.7 DeltaNet: Deferred Proofs . . . 308

T Analysis of Mamba 326
T.1 Definition of Mamba . . . 326
T.2 Mamba models are smooth unitwise-contractive . . . 327
T.3 Positive-margin expressivity of Mamba . . . 329

U Analysis of mLSTM 330
U.1 Definition of mLSTM . . . 330

U.2 Positive-Margin Expressivity 331

Part I

Background

A Extended Background

Table of Contents

A.1 Notation . . . 23
A.2 Functions . . . 23
A.3 Sequences . . . 26
A.4 Vectors and Matrices . . . 28
A.5 Notation for Derivatives . . . 29
A.6 Partitions of a Set . . . 32
A.7 Affine Functions . . . 33
A.8 Piecewise Representations of Functions . . . 33
A.9 Piecewise-Affine Functions . . . 34
A.10 Analytic Euclidean Maps and Neural Primitives . . . 37
A.11 Affine Hyperplanes and Semilinear Sets . . . 40
A.12 PAP Functions and Intensional Derivatives . . . 42
A.13 Lattice Theory . . . 44
A.14 Algebraic Structures . . . 45
A.15 Semigroup and Group Theory . . . 47
A.16 Algebraic Automata Theory . . . 49
A.17 Metric Spaces and Ambient Spaces . . . 50
A.18 Measure Theory . . . 52
A.19 Metric Automata Theory . . . 54
A.20 Definite Languages . . . 55
A.21 Graphs, Isomorphisms, and Renaming of Nodes . . . 56

A.1 Notation

We write $\mathbb{N}$ for the natural numbers, we write $\mathbb{N}^{>0}$ for the natural numbers exlcuding zero, we write $\mathbb{R}$ for the real numbers, we write $\mathbb{R}^{>0}$ for the non-negative real numbers, and we write $\mathbb{R}^{>0}$ for the positive real numbers—zero is excluded.

We write $\mathbb{T}$ for $\mathbb{N}^{>0}$ when it is meant as the set of all timepoints.

For X a set, we write $\mathcal{P}_+(X)$ for the set of all non-empty subsets of X.

Given $i, j \in \mathbb{N}$, we write $[i..j]$ for the integer interval $\{i, \ldots, j\}$, we write $(i..j]$ for the integer interval $\{i+1, \ldots, j\}$, and we write $[i..j)$ for the integer interval $\{i, \ldots, j-1\}$. Such intervals can be empty, e.g., the interval $(1..1]$ is empty.

For $(X, \|\cdot\|)$ a normed space and for $r \in \mathbb{R}^{\geq 0}$, we define $X_{\leq r} := \{x \in X \mid \|x\| \leq r\}$.

A.2 Functions

Definition 5 (Injective, surjective, and bijective functions)**.** A function $f : X \to Y$ is:

- *injective* if, for every $x, x' \in X$,
$$f(x) = f(x') \quad \Longrightarrow \quad x = x';$$
- *surjective* if, for every $y \in Y$, there exists $x \in X$ such that $f(x) = y$;
- *bijective* if it is both injective and surjective—also called a *bijection*.

End of Definition 5.

Fact 1 (Inverse of a bijection)**.** *A function $f : X \to Y$ is bijective if and only if there exists a unique function*

$$f^{-1} : Y \to X$$

such that

$$f^{-1} \circ f = \mathrm{id}_X,$$
$$f \circ f^{-1} = \mathrm{id}_Y,$$

where id_X and id_Y denote the identity functions on X and Y, respectively. The function f^{-1} is called the inverse *of f.* End of Fact 1.

Definition 6 (Identity function)**.** The *identity function* on a set X is the function

$$\mathrm{id}_X : X \to X,$$
$$x \mapsto x.$$

We write id in place of id_X when X is clear from the context. *End of Definition 6.*

Definition 7 (Transformation and permutation)**.** A *transformation* on a set X is a function $f : X \to X$. A *permutation* on a set X is a bijective transformation on X. *End of Definition 7.*

Fact 2. *Every identity function id_X is a permutation on X.* End of Fact 2.

We use the following standard operations on functions.

Definition 8 (Product of functions)**.** Let

$$f : X \to Y, \qquad g : U \to V.$$

The *product* of f and g is the function

$$f \times g : X \times U \to Y \times V$$

defined by

$$(f \times g)(x, u) := (f(x), g(u)).$$

More generally, for functions

$$f_i : X_i \to Y_i, \qquad i \in [1..k],$$

their product is the function

$$f_1 \times \cdots \times f_k : X_1 \times \cdots \times X_k \to Y_1 \times \cdots \times Y_k$$

defined by

$$\big(f_1 \times \cdots \times f_k\big)(x_1, \ldots, x_k) := (f_1(x_1), \ldots, f_k(x_k)).$$

End of Definition 8.

Definition 9 (Tupling of functions)**.** Let

$$f_i : X \to Y_i, \qquad i \in [1..k],$$

be functions having a common domain. Their *tupling* is the function

$$(f_1, \ldots, f_k) : X \to Y_1 \times \cdots \times Y_k$$

defined by

$$(f_1, \ldots, f_k)(x) := (f_1(x), \ldots, f_k(x)).$$

End of Definition 9.

Remark 1. The product and tupling operations are distinct. In particular,

$$f_1 \times \cdots \times f_k$$

acts on a tuple of independent arguments, whereas

$$(f_1, \ldots, f_k)$$

evaluates all functions on the same argument. *End of Remark 1.*

Definition 10 (Function composition)**.** Let

$$f : X \to Y, \qquad g : Y \to Z.$$

The *composition* of f followed by g is the function

$$g \circ f : X \to Z$$

defined by

$$(g \circ f)(x) := g\big(f(x)\big).$$

More generally, for composable functions

$$f_i : X_{i-1} \to X_i, \qquad i \in [1..k],$$

we write

$$f_k \circ \cdots \circ f_1 : X_0 \to X_k$$

for their iterated composition, defined by

$$\big(f_k \circ \cdots \circ f_1\big)(x) := f_k\big(\cdots f_2\big(f_1(x)\big)\cdots\big).$$

End of Definition 10.

Remark 2. Function composition is associative whenever the corresponding functions are composable. Hence parentheses in iterated compositions are omitted. *End of Remark 2.*

Definition 11 (Closure of class of functions)**.** For $\mathcal{P}$ a class of functions, its *closure* $\mathrm{cl}(\mathcal{P})$ is the smallest class of functions such that:

- $\mathcal{P} \subseteq \mathrm{cl}(\mathcal{P})$;
- $\mathrm{cl}(\mathcal{P})$ is closed under function tupling:

$$\big\{(f_1, \ldots, f_n) \;\big|\; f_1, \ldots, f_n \in \mathrm{cl}(\mathcal{P}),\, n \in \mathbb{N}^{>0}\big\} \;\subseteq\; \mathcal{F};$$

- $\mathrm{cl}(\mathcal{P})$ is closed under function composition:

$$\big\{f \circ g \;\big|\; f : \mathbb{R}^{m_1} \to \mathbb{R}^m \ldots, f_n \in \mathrm{cl}(\mathcal{P}),\, n \in \mathbb{N}^{>0}\big\} \;\subseteq\; \mathrm{cl}(\mathcal{P}).$$

End of Definition 11.

A.2.1 Euclidean Functions

Definition 12. A *function between finite-dimensional Euclidean spaces* is a function $f : X \to Y$ where $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ for some $n \in \mathbb{N}$ and some $m \in \mathbb{N}^{>0}$.

We also call f a *Euclidean function* or *Euclidean map*.

The following terminology applies:

- X is the *input domain* or *input space*;
- Y is the *codomain* or *output space*;
- $\mathrm{Im}(f) := \{f(x) \mid x \in X\}$ is the *image*;
- $\mathrm{ar}_{\mathrm{in}}(f) := n$ is the *input dimension* or *input arity*;
- $\mathrm{ar}_{\mathrm{out}}(f) := m$ is the *output dimension* or *output arity*.

End of Definition 12.

Definition 13. A *parametric Euclidean function* is a function

$$f : X \times \Theta \to Y,$$

where

$$X \subseteq \mathbb{R}^n, \qquad \Theta \subseteq \mathbb{R}^p, \qquad Y \subseteq \mathbb{R}^m,$$

for some $n, p \in \mathbb{N}$ and some $m \in \mathbb{N}^{>0}$.

We typically write $f(x; \theta)$ in place of $f(x, \theta)$.

The following terminology applies:

- X is the *input domain* or *input space*;
- Θ is the *parameter domain* or *parameter space*;
- Y is the *codomain* or *output space*;
- $\mathrm{ar}_{\mathrm{in}}(f) := n$ is the *input dimension* or *input arity*;
- $\mathrm{ar}_{\mathrm{out}}(f) := p$ is the *parameter dimension* or *parameter arity*;
- $\mathrm{ar}_{\mathrm{out}}(f) := m$ is the *output dimension* or *output arity*.

For every $\theta \in \Theta$, the function

$$f[\theta] : X \to Y, \qquad f[\theta](x) := f(x; \theta),$$

is called the *instance of f with parameter values θ*. *End of Definition 13.*

Fact 3. *We consider every parametric Euclidean function $f : X \times \Theta \to Y$ to also be a Euclidean function. Indeed, if $X \subseteq \mathbb{R}^n$, $\Theta \subseteq \mathbb{R}^p$, and $Y \subseteq \mathbb{R}^m$, then the joint input-parameter space $X \times \Theta$ can be identified with a subset of $\mathbb{R}^{n+p}$.*

A.3 Sequences

A.3.1 Finite Sequences

Definition 14 (Finite sequences)**.** Let X be a set and let $n \in \mathbb{N}$.

A *finite sequence x over X* written *in explicit form* is an expression

$$x_{i_1}, \ldots, x_{i_n}$$

where

- $n \in \mathbb{N}$ is called the *length* of x;
- $i_1, \ldots, i_n \in \mathbb{N}$ are indices satisfying $i_1 < \ldots < i_n$, and we call $I = \{i_1, \ldots, i_n\}$ the index set of x
- for every $j \in [1..n]$, we have $x_{i_j} \in X$, called the *j-th element* of x.

Such sequence x can be equivalently written in *indexed form* as

$$(x_i)_{i \in I},$$

and the following terminology applies:

- x is an *empty sequence* if $|x| = 0$;
- x is a *non-empty sequence* if $|x| > 0$.

End of Definition 14.

Fact 4. *There exists a unique empty sequence, regardless of the choice of an underlying set X. Such sequence is therefore called* the empty sequence*, and it is denoted by ε. Thus, instead of saying that a sequence x is* an empty sequence*, we can say that x is* the empty sequence. End of Fact 4.

Definition 15 (Sets of finite sequences)**.** Let X be a set, and let $N \in \mathbb{N}^{>0}$.

We define the following notation for sets of finite sequences:

- X^* denotes the set of all finite sequences over X;
- X^+ denotes the set of all finite *non-empty* sequences over X;
- X^N denotes the set of all sequences over X having length N;
- $X^{*\leq N}$ denotes the set consisting of each sequence x over X having length $|x| \leq N$;

- $X^{+\leq N}$ denotes the set consisting of each non-empty sequence x over X having length $|x| \leq N$.

End of Definition 15.

Definition 16 (Slicing operator)**.** For $x = x_{i_1}, \ldots, x_{i_n}$ a finite sequence over a set X, and for $j, \ell \in [1..n]$ with $j \leq \ell$, we define

$$x_{[i_j \,..\, i_\ell]} := x_{i_j}, x_{i_j+1}, \ldots, x_{i_\ell}$$

which is itself a finite sequence over X having length $\ell - j + 1$. *End of Definition 16.*

A.3.2 Infinite Sequences

Definition 17 (Infinite sequences)**.** Let X be a set.

An *infinite sequence x over X* written in *explicit form* is an expression

$$x_{i_1}, \ldots, x_{i_n}, \ldots$$

where:

- for every $j \in \mathbb{N}^{>0}$, we have $i_j \in \mathbb{N}$;
- for every $j, \ell \in \mathbb{N}^{>0}$, we have that $j < \ell$ implies $i_j < i_\ell$;
- for every $j \in \mathbb{N}^{>0}$, we have $x_{i_j} \in X$, called the *j-th element* of x.

We call $I = \{i_j \in \mathbb{N} \mid j \in \mathbb{N}^{>0}\}$ the *index set* of x.

The above sequence x can be equivalently written *in indexed form* as

$$(x_{i_j})_{j \in \mathbb{N}^{>0}},$$

and also as

$$(x_i)_{i \in I}.$$

End of Definition 17.

Definition 18 (Sets of finite sequences)**.** Let X be a set.

We define the following notation for sets of infinite sequences:

- X^ω denotes the set of all infinite sequences over X;
- when X is equipped with a norm $\|\cdot\|_X$, we write $X^{\omega \mathrm{b}}$ for the set of all bounded infinite sequences over X; namely $X^{\omega \mathrm{b}}$ consists of each sequence $x = (x_i)_{i \in I} \in X^\omega$ satisfying

  $$\sup_{i \in I} \|x_i\|_X \;\leq\; B_x$$

  for some constant $B_x \in \mathbb{R}^{\geq 0}$.

End of Definition 18.

Definition 19 (Slicing operator)**.** For $x = (x_i)_{i \in I}$ an infinite sequence over a set X, and for $j, \ell \in I$ satisfying $j \leq \ell$, we define

$$x_{[i \,..\, \ell]} := x_i, x_{i+1}, \ldots, x_\ell$$

which is a finite sequence over X having length $\ell - i + 1$. *End of Definition 19.*

A.3.3 String Notation

A *string* is a sequence, with the only notational difference that commas are omitted when it is written in explict form. For instance,

$$x_1, x_2 \ldots, x_n \qquad \rightsquigarrow \qquad x_1 x_2 \cdots x_n$$

and

$$x_1, x_2 \ldots, x_n, \ldots \qquad \rightsquigarrow \qquad x_1 x_2 \cdots x_n \cdots$$

for finite and infinite strings, respectively.

A.4 Vectors and Matrices

Let $\mathbb{D}$ be a set.

For $n \in \mathbb{N}_{>0}$, a *vector of dimension n over $\mathbb{D}$* is an element $x \in \mathbb{D}^n$. We write vectors as

$$x = (x_1, \ldots, x_n) = \big(x_1, \ldots, x_n\big),$$

where, for every $i \in \{1, \ldots, n\}$,

$$(x)_i = x_i \in \mathbb{D}$$

is called the *i-th component* of x.

For $m, n \in \mathbb{N}_{>0}$, a *matrix of dimension $m \times n$ over $\mathbb{D}$* is an element $A \in \mathbb{D}^{m \times n}$. We write matrices as

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix},$$

where, for every $i \in [1..m]$ and $j \in [1..n]$,

$$(A)_{i,j} = a_{i,j} \in \mathbb{D}$$

is called the *(i, j)-th entry* of A. For every $i \in [1..m]$, we denote the *i-th row* of A by

$$(A)_{i,:} := \begin{pmatrix} a_{i,1} & \ldots & a_{i,n} \end{pmatrix} \in \mathbb{D}^{1 \times n}$$

We use *entry* as a generic term for both a component of a vector and an entry of a matrix.

Whenever vectors are used in a matrix setting, we identify them with column vectors. Thus, for $x = (x_1, \ldots, x_n) \in \mathbb{D}^n$, we identify

$$x \equiv \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{D}^{n \times 1}.$$

For instance, matrix–vector products such as $A \cdot x$, for $A \in \mathbb{D}^{m \times n}$, are understood with x interpreted as a column vector.

Definition 20 (Set of entries)**.** Let $\mathbb{D}$ be a domain.

For a vector $x \in \mathbb{D}^n$, we define its *set of entries* as

$$\operatorname{entries}(x) := \{(x)_i \mid i \in [1..n]\}.$$

For a matrix $A \in \mathbb{D}^{m \times n}$, we define its *set of entries* as

$$\operatorname{entries}(A) := \{(A)_{i,j} \mid i \in [1..m],\ j \in [1..n]\}.$$

Definition 21 (Deterministic row-substochastic matrix)**.** A matrix $A \in \mathbb{R}^{n \times n}$ is *deterministic row-substochastic* if

$$A \in \{0, 1\}^{n \times n}, \qquad A \cdot 1_n \leq 1_n.$$

Equivalently, every row is either zero or a standard basis row. In particular, $A \geq 0_{n,n}$ and $\|A\|_\infty \leq 1$.

End of Definition 21.

A.4.1 Norms

We introduce norms of vectors and matrices over $\mathbb{R}$.

Definition 22. The *Euclidean norm* of a vector $x \in \mathbb{R}^n$ is

$$\|x\| := \sqrt{\sum_{i=1}^{n} \big((x)_i\big)^2}.$$

End of Definition 22.

Definition 23. The *maximum norm* of a vector $x \in \mathbb{R}^n$ is

$$\|x\|_\infty := \|x\|_{\max} := \max_{1 \le i \le n} \big|(x)_i\big|.$$

End of Definition 23.

Fact 5 (Relationship between Euclidean and maximum norms)**.** *The following inequality holds:*

$$\|x\| \le \sqrt{n} \cdot \|x\|_\infty = \sqrt{n} \cdot \|x\|_{\max} \qquad \forall x \in \mathbb{R}^n.$$

End of Fact 5.

Definition 24 (Spectral norm)**.** The *spectral norm* of a matrix $A \in \mathbb{R}^{m \times n}$ is

$$\|A\|_2 := \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|A \cdot x\|}{\|x\|}.$$

When we refer to the *norm* of a matrix, we mean its *spectral norm*; and we may write $\|A\|$ in place of $\|A\|_2$. *End of Definition 24.*

Definition 25 (Frobenius norm)**.** The *Frobenius norm* of a matrix $A \in \mathbb{R}^{m \times n}$ is

$$\|A\|_{\mathrm{F}} := \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \big((A)_{i,j}\big)^2}.$$

End of Definition 25.

Fact 6 (Relationship between Frobenius and spectral norms)**.** *For every $A \in \mathbb{R}^{m \times n}$,*

$$\|A\| \;\le\; \|A\|_{\mathrm{F}} \;\le\; \sqrt{\operatorname{rank}(A)} \cdot \|A\| \;\le\; \sqrt{\min\{m, n\}} \cdot \|A\|,$$

where $\|A\|$ denotes the spectral norm of A. End of Fact 6.

Definition 26 (Maximum and row norms)**.** The *maximum norm* of a matrix $A \in \mathbb{R}^{m \times n}$ is

$$\|A\|_{\max} := \max_{1 \le i \le m} \max_{1 \le j \le n} \big|(A)_{i,j}\big|.$$

The *row-wise maximum Euclidean norm*, or simply *maximum norm*, of a matrix $A \in \mathbb{R}^{m \times n}$ is

$$\|A\|_{\mathrm{row}} := \max_{i \in [1..m]} \|(A)_{i,:}\|.$$

End of Definition 26.

A.5 Notation for Derivatives

We use D to denote the derivative operator on functions between finite-dimensional Euclidean spaces.

Let

$$f : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}}$$

be differentiable. We denote its derivative by

$$\mathsf{D}f : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}},$$

where

$$\mathsf{D}f(x)$$

is the Jacobian matrix of f at $x \in \mathbb{R}^{d_{\mathrm{in}}}$.

A.5.1 Positional partial derivatives

Let us consider a Euclidean function $f : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}}$.

Let $x \in \mathbb{R}^{d_{\mathrm{in}}}$ be an input of f, which we write as

$$x = (x_1, \ldots, x_{d_{\mathrm{in}}}).$$

For every $i \in [1..d_{\mathrm{in}}]$, we denote by

$$\mathsf{D}_i f$$

the *partial derivative of f with respect to its i-th input coordinate*. Thus,

$$\mathsf{D}_i f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}},$$

and

$$\mathsf{D}_i f(x)$$

is the i-th column of $\mathsf{D}f(x)$, identified with a vector in $\mathbb{R}^{d_{\text{out}}}$.

Accordingly,

$$\mathsf{D}f(x) = \begin{pmatrix} \mathsf{D}_1 f(x) & \cdots & \mathsf{D}_{d_{\text{in}}} f(x) \end{pmatrix}.$$

A.5.2 Blockwise positional partial derivatives

Let $f : X \to Y$ be a Euclidean function where

$$X = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_k}, \qquad Y = \mathbb{R}^{d_{\text{out}}},$$

and let $x \in X$ be an input of f.

We canonically identify

$$X \cong \mathbb{R}^{d_1 + \cdots + d_k},$$

and accordingly write

$$x = (x_1, \ldots, x_{d_1 + \cdots + d_k}).$$

For every $i \in [1..k]$, let

$$b_i := \sum_{r=1}^{i-1} d_r,$$

and define the *i-th block-index vector* by

$$[i] := (b_i + 1, \ldots, b_i + d_i).$$

Thus, $[i]$ is the ordered vector of scalar input-coordinate indices belonging to the i-th input block.

We denote by

$$x_{[i]} := (x_{b_i+1}, \ldots, x_{b_i+d_i}) \in \mathbb{R}^{d_i}$$

the subvector of x whose coordinates are indexed by $[i]$, and call it the *i-th input block* of x.

For every $i \in [1..k]$, we denote by

$$\mathsf{D}_{[i]} f$$

the *partial derivative of f with respect to its i-th input block*. Thus,

$$\mathsf{D}_{[i]} f : X \to \mathbb{R}^{d_{\text{out}} \times d_i}.$$

More precisely, for every $x \in X$, we define

$$\mathsf{D}_{[i]} f(x) := \begin{pmatrix} \mathsf{D}_{b_i+1} f(x) & \cdots & \mathsf{D}_{b_i+d_i} f(x) \end{pmatrix},$$

where $\mathsf{D}_j f$ denotes the partial derivative of f with respect to its j-th scalar input coordinate, as defined in §A.5.1.

Equivalently, $\mathsf{D}_{[i]} f(x)$ is the submatrix of $\mathsf{D}f(x)$ formed, in their original order, by the columns whose indices belong to $[i]$.

Consequently,

$$\mathsf{D}f(x) = \begin{pmatrix} \mathsf{D}_{[1]} f(x) & \cdots & \mathsf{D}_{[k]} f(x) \end{pmatrix}.$$

A.5.3 Named partial derivatives

Let us consider a Euclidean function

$$f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}},$$

and let

$$x = (x_1, \ldots, x_{d_{\text{in}}}) \in \mathbb{R}^{d_{\text{in}}}.$$

When evaluating a positional partial derivative at x, we also use a named partial-derivative notation in which the subscript identifies the corresponding scalar input coordinate of the evaluation point.

More precisely, for every $i \in [1..d_{\text{in}}]$, we write

$$\mathsf{D}_{x_i} f\big(x_1, \ldots, x_{d_{\text{in}}}\big) := \mathsf{D}_{[i]} f\big(x_1, \ldots, x_{d_{\text{in}}}\big).$$

Thus,

$$\mathsf{D}_{x_i} f\big(x_1, \ldots, x_{d_{\text{in}}}\big) \in \mathbb{R}^{d_{\text{out}}}$$

denotes the partial derivative of f at x with respect to the scalar input coordinate x_i.

Hence, $\mathsf{D}_{[i]} f$ is the positional partial-derivative function associated with the i-th scalar input coordinate, whereas

$$\mathsf{D}_{x_i} f\big(x_1, \ldots, x_{d_{\text{in}}}\big)$$

is notation for that partial derivative evaluated at x and named according to the corresponding scalar input coordinate.

When a scalar input coordinate at the evaluation point is itself given by the value of a function or expression, we often identify that coordinate by the corresponding function or expression symbol rather than by its full value.

More precisely, suppose that

$$f : \mathbb{R}^2 \to \mathbb{R}^{d_{\text{out}}},$$

and consider an evaluation point of the form

$$(g(x), y),$$

where $g(x), y \in \mathbb{R}$. The fully explicit named partial-derivative notation for the first scalar input coordinate is

$$\mathsf{D}_{g(x)} f\big(g(x), y\big).$$

When there is no ambiguity, we abbreviate the name of this scalar input coordinate by the symbol of the function producing it, and write

$$\mathsf{D}_{g} f\big(g(x), y\big) := \mathsf{D}_{g(x)} f\big(g(x), y\big).$$

More generally, named subscripts may abbreviate the expression defining the corresponding scalar input coordinate whenever the association between the name and the coordinate at the evaluation point is clear from context.

A.5.4 Blockwise named partial derivatives

Let $f : X \to Y$ be a Euclidean function where

$$X = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_k}, \qquad Y = \mathbb{R}^{d_{\text{out}}},$$

and let

$$x = (x_1, \ldots, x_{d_1 + \cdots + d_k}) \in X,$$

where, as in the preceding subsection, we canonically identify

$$X \cong \mathbb{R}^{d_1 + \cdots + d_k}.$$

For every $i \in [1..k]$, recall that $[i]$ is the i-th block-index vector and

$$x_{[i]} \in \mathbb{R}^{d_i}$$

is the corresponding i-th input block of x.

When evaluating a blockwise positional partial derivative at x, we also use a named partial-derivative notation in which the subscript identifies the corresponding input block of the evaluation point.

More precisely, for every $i \in [1..k]$, we write

$$\mathsf{D}_{x_{[i]}} f(x) := \mathsf{D}_{[i]} f(x).$$

Thus,

$$\mathsf{D}_{x_{[i]}} f(x) \in \mathbb{R}^{d_{\text{out}} \times d_i}$$

denotes the Jacobian block of f at x corresponding to variations of the input block $x_{[i]}$, that is, to variations of the scalar input coordinates whose indices belong to $[i]$.

Hence, $\mathsf{D}_{[i]} f$ is the blockwise positional partial-derivative function associated with the i-th input block, whereas

$$\mathsf{D}_{x_{[i]}} f(x)$$

is notation for that partial derivative evaluated at x and named according to the corresponding input block.

For example, if

$$f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_{\text{out}}},$$

then, at a point

$$(x, \theta),$$

we write

$$\mathsf{D}_x f(x, \theta) \in \mathbb{R}^{d_{\text{out}} \times d_x} \qquad \text{and} \qquad \mathsf{D}_\theta f(x, \theta) \in \mathbb{R}^{d_{\text{out}} \times d_\theta}$$

for the Jacobian blocks corresponding to the x and θ input blocks, respectively.

When an input block at the evaluation point is itself given by the value of a function or expression, we often identify that input block by the corresponding function or expression symbol rather than by its full value.

More precisely, suppose that

$$f : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_{\text{out}}},$$

and consider an evaluation point of the form

$$(g(x), y),$$

where

$$g(x) \in \mathbb{R}^{d_1}, \qquad y \in \mathbb{R}^{d_2}.$$

The fully explicit named partial-derivative notation for the first input block is

$$\mathsf{D}_{g(x)} f\big(g(x), y\big).$$

When there is no ambiguity, we abbreviate the name of this input block by the symbol of the function producing it, and write

$$\mathsf{D}_g f\big(g(x), y\big) := \mathsf{D}_{g(x)} f\big(g(x), y\big).$$

More generally, named subscripts may abbreviate the expression defining the corresponding input block whenever the association between the name and the input block at the evaluation point is clear from context.

A.6 Partitions of a Set

Definition 27 (Partition)**.** Let X be a set. A *partition* of X is a set $\mathcal{P}$ of nonempty subsets of X such that

$$\bigcup_{P \in \mathcal{P}} P = X,$$

and

$$P \cap P' = \emptyset \qquad \forall P, P' \in \mathcal{P} \text{ with } P \neq P'.$$

The elements of $\mathcal{P}$ are called the *regions* of the partition $\mathcal{P}$, or $\mathcal{P}$*-regions*.

Definition 28 (Refinement of a partition)**.** Let $\mathcal{P}$ and $\mathcal{Q}$ be partitions of a set X. We say that $\mathcal{Q}$ is a *refinement* of $\mathcal{P}$, denoted by

$$\mathcal{Q} \preceq \mathcal{P},$$

if, for every $Q \in \mathcal{Q}$, there exists $P \in \mathcal{P}$ such that

$$Q \subseteq P,$$

where P is called the corresponding $\mathcal{P}$-region for Q, which we write as $\mathcal{P}[Q] := P$, and as $\mathcal{P}[q] := P$ when $Q = \{q\}$.

A.7 Affine Functions

Definition 29 (Affine and linear functions)**.** Let $X \subseteq \mathbb{R}^d$ and let $Y \subseteq \mathbb{R}^m$. A function $g : X \to Y$ is *affine* if there exist a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $b \in \mathbb{R}^m$ such that

$$g(x) \;=\; A \cdot x + b \qquad\qquad \forall x \in X;$$

in this case, we refer to $\langle A, b\rangle$ as *affine coefficients* for g.

An affine function g is *linear* if it admits affine coefficients of the form $\langle A, 0_m\rangle$; equivalently,

$$g(x) \;=\; A \cdot x \qquad\qquad \forall x \in \mathbb{R}^d.$$

A.8 Piecewise Representations of Functions

Definition 30 (Piecewise representation)**.** Let X and Y be spaces.

Syntax. A *piecewise representation* on (X, Y) is a set

$$\gamma = \big\{\langle P, f_P\rangle\big\}_{P \in \mathcal{P}},$$

such that

- $\mathcal{P}$ is a finite partition of X, and
- $f_P : P \to Y$ for every $P \in \mathcal{P}$;

the following terminology applies:

- $\mathcal{P}$ is *the (input-)partitition of* γ;
- every $P \in \mathcal{P}$ is *a region of* γ;
- every function f_P for $P \in \mathcal{P}$ is *the piece of* γ *on the region* P.

Semantics. The *function represented by* γ is[1]

$$\langle\!\langle \gamma \rangle\!\rangle : X \to Y, \qquad\qquad \langle\!\langle \gamma \rangle\!\rangle(x) := f_{\mathcal{P}[x]}(x) \qquad \forall x \in X.$$

Then, for f a function, we say that γ is a *piecewise representation of* f if $\langle\!\langle \gamma \rangle\!\rangle = f$. In this case, the following terminology applies:

- $\mathcal{P}$ is *the (input-)partitition of* f *when represented by* γ;
- every $P \in \mathcal{P}$ is *a region of* f *when represented by* γ;
- every function f_P for $P \in \mathcal{P}$ is *the piece of* f *on the region* P *when represented by* γ.

End of Definition 30.

[1] Recall the notation $\mathcal{P}[x]$ for $x \in X$ introduced in §A.6.

A.9 Piecewise-Affine Functions

Definition 31 (Piecewise-affine representation)**.** A *piecewise representation*

$$\gamma = \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}},$$

is affine if, for every $P \in \mathcal{P}$, the piece f_P is an affine function.

Definition 32 (Piecewise-affine function)**.** A function f is *piecewise affine* if there exists a piecewise-affine representation γ that represents f.

Definition 33 (Piecewise-affine coefficient-based representation)**.** Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$.

Syntax. A *piecewise-affine coefficient-based representation* on (X, Y) is a set

$$\gamma = \big\{\langle P, A_P, b_P\rangle\big\}_{P\in\mathcal{P}},$$

such that: (i) $\mathcal{P}$ is a finite partition of X; and (ii) $A_P \in \mathbb{R}^{m\times d}$ and $b_P \in \mathbb{R}^m$, for every $P \in \mathcal{P}$. The following terminology applies:

- $\mathcal{P}$ is *the (input-)partitition of* γ;
- every $P \in \mathcal{P}$ is *a region of* γ;
- $\langle A_P, b_P\rangle$ for $P \in \mathcal{P}$ are *the affine coefficients of* γ *on the region* P.

Semantics. The *function represented by* γ is[2]

$$\langle\!\langle \gamma \rangle\!\rangle(x) := A_{\mathcal{P}x} \cdot x + b_{\mathcal{P}x} \qquad \forall x \in X.$$

End of Definition 33.

Fact 7. *We state the following two facts on the interchangeability of piecewise-affine representations and their coefficient-based counterpart.*

Fact 7.1. *For every piecewise representation* γ *on* (X, Y) *with*

$$\gamma = \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}}, \quad X \subseteq \mathbb{R}^d, \quad Y \subseteq \mathbb{R}^m,$$

the following two claims are equivalent:

- γ *is piecewise affine;*
- *there exists a piecewise-affine coefficient-based representation* $\hat{\gamma}$ *on* (X, Y) *such that*

  $$\hat{\gamma} = \big\{\langle P, A_P, b_P\rangle\big\}_{P\in\mathcal{P}} \qquad \textit{with} \qquad f_P(x) \;=\; A_P \cdot x + b_P \qquad \forall P \in \mathcal{P}, \forall x \in X;$$

  in particular, we have $\langle\!\langle \gamma \rangle\!\rangle = \langle\!\langle \hat{\gamma} \rangle\!\rangle$.

Fact 7.2. *For every piecewise-affine coefficient-based representation*

$$\hat{\gamma} = \big\{\langle P, A_P, b_P\rangle\big\}_{P\in\mathcal{P}},$$

the piecewise-affine representation

$$\gamma := \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}} \qquad \textit{having} \qquad f_P(x) \;:=\; A_P \cdot x + b_P \qquad \forall P \in \mathcal{P}, \forall x \in X$$

satisfies

$$\langle\!\langle \gamma \rangle\!\rangle = \langle\!\langle \hat{\gamma} \rangle\!\rangle.$$

We refer to γ *as the* (canonical) piecewise-affine representation *induced by* $\hat{\gamma}$. End of Fact 7.

Definition 34 (Piecewise-linear representation)**.** A piecewise-affine representation

$$\gamma = \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}},$$

is *piecewise linear* if, for every $P \in \mathcal{P}$, the piece f_P is a linear map.

[2] Recall the notation $\mathcal{P}[x]$ for $x \in X$ introduced in §A.6.

Fact 8. *We state the following two facts on the interchangeability of piecewise-linear representations and their coefficient-based counterpart.*

Fact 8.1. *For every γ a piecewise representation on (X, Y) with*

$$\gamma = \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}}, \quad X \subseteq \mathbb{R}^d, \quad Y \subseteq \mathbb{R}^m,$$

the following two claims are equivalent:

- *γ is piecewise linear;*
- *there exists a piecewise-affine coefficient-based representation $\hat{\gamma}$ on (X, Y) such that*

$$\hat{\gamma} = \big\{\langle P, A_P, 0_m\rangle\big\}_{P\in\mathcal{P}} \qquad \textit{with} \qquad f_P(x) \;=\; A_P \cdot x \qquad \forall P \in \mathcal{P}, \forall x \in X;$$

in particular, we have $\langle\!\langle\gamma\rangle\!\rangle = \langle\!\langle\hat{\gamma}\rangle\!\rangle$.

Fact 8.2. *For every piecewise-affine coefficient-based representation*

$$\hat{\gamma} = \big\{\langle P, A_P, 0_m\rangle\big\}_{P\in\mathcal{P}},$$

the piecewise-linear representation

$$\gamma := \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}} \qquad \textit{having} \qquad f_P(x) \;:=\; A_P \cdot x \qquad \forall P \in \mathcal{P}, \forall x \in X$$

satisfies

$$\langle\!\langle\gamma\rangle\!\rangle = \langle\!\langle\hat{\gamma}\rangle\!\rangle.$$

We refer to γ as the (canonical) piecewise-linear representation *induced by $\hat{\gamma}$.* End of Fact 8.

Definition 35 (Partially-affine piecewise representations and functions)**.** Let $X \subseteq \mathbb{R}^n$, $U \subseteq \mathbb{R}^d$, and $Y \subseteq \mathbb{R}^m$.

A piecewise representation

$$\gamma = \big\{\langle P, f_P\rangle\big\}_{P\in\mathcal{P}}$$

on $(X \times U, Y)$ is *affine in the first argument* if, for every $P \in \mathcal{P}$, there exist a matrix

$$A_P \in \mathbb{R}^{m\times n}$$

and a function

$$b_P : U \to \mathbb{R}^m$$

such that

$$f_P(x, u) \;=\; A_P \cdot x + b_P(u) \qquad \forall (x, u) \in X \times U.$$

In this case, we call $\langle A_P, b_P\rangle$ a *coefficient-offset pair of γ on the region P*; in particular, A_P is the *linear coefficient* and b_P is the *offset function* of γ on P.

We say that a function

$$f : X \times U \to Y$$

is *piecewise affine in its first argument* if it is represented by some piecewise representation that is affine in the first argument. *End of Definition 35.*

Definition 36 (Partially-affine coefficient-based piecewise representation)**.** Let $X \subseteq \mathbb{R}^n$, $U \subseteq \mathbb{R}^d$, and $Y \subseteq \mathbb{R}^m$.

Syntax. A *coefficient-based piecewise representation on (X, U, Y) that is affine in the first argument* is a set

$$\hat{\gamma} = \big\{\langle P, A_P, b_P\rangle\big\}_{P\in\mathcal{P}},$$

such that:

(1) $\mathcal{P}$ is a finite partition of $X \times U$;

(2) $A_P \in \mathbb{R}^{m\times n}$ for every $P \in \mathcal{P}$;

(3) $b_P : U \to \mathbb{R}^m$ for every $P \in \mathcal{P}$.

The following terminology applies:

- $\mathcal{P}$ is *the (input-)partition of* $\hat{\gamma}$;
- every $P \in \mathcal{P}$ is *a region of* $\hat{\gamma}$;
- $\langle A_P, b_P \rangle$ is *the coefficient-offset pair of* $\hat{\gamma}$ *on the region* P;
- A_P is *the linear coefficient of* $\hat{\gamma}$ *on the region* P;
- b_P is *the offset function of* $\hat{\gamma}$ *on the region* P.

Semantics. The *function represented by* $\hat{\gamma}$ is

$$\langle\!\langle \hat{\gamma} \rangle\!\rangle(x, u) := A_{\mathcal{P}(x,u)} \cdot x + b_{\mathcal{P}(x,u)}(u) \qquad \forall (x, u) \in X \times U.$$

End of Definition 36.

Fact 9. *We state the following two facts on the interchangeability of piecewise representations that are affine in the first argument and their coefficient-based counterparts.*

Fact 9.1. *For every piecewise representation* γ *on* $(X \times U, Y)$ *with*

$$\gamma = \left\{ \langle P, f_P \rangle \right\}_{P \in \mathcal{P}}, \quad X \subseteq \mathbb{R}^n, \quad U \subseteq \mathbb{R}^d, \quad Y \subseteq \mathbb{R}^m,$$

the following two claims are equivalent:

- γ *is affine in the first argument;*
- *there exists a coefficient-based piecewise representation* $\hat{\gamma}$ *on* (X, U, Y) *that is affine in the first argument such that*

  $$\hat{\gamma} = \left\{ \langle P, A_P, b_P \rangle \right\}_{P \in \mathcal{P}} \qquad \textit{with} \qquad f_P(x, u) = A_P \cdot x + b_P(u)$$

  for every $P \in \mathcal{P}$ *and every* $(x, u) \in X \times U$*;*

  in particular,

  $$\langle\!\langle \gamma \rangle\!\rangle = \langle\!\langle \hat{\gamma} \rangle\!\rangle.$$

Fact 9.2. *For every coefficient-based piecewise representation* $\hat{\gamma}$ *on* (X, U, Y) *that is affine in the first argument,*

$$\hat{\gamma} = \left\{ \langle P, A_P, b_P \rangle \right\}_{P \in \mathcal{P}},$$

the piecewise representation

$$\gamma := \left\{ \langle P, f_P \rangle \right\}_{P \in \mathcal{P}} \qquad \textit{having} \qquad f_P(x, u) := A_P \cdot x + b_P(u)$$

for every $P \in \mathcal{P}$ *and every* $(x, u) \in X \times U$ *is affine in the first argument and satisfies*

$$\langle\!\langle \gamma \rangle\!\rangle = \langle\!\langle \hat{\gamma} \rangle\!\rangle.$$

We refer to γ *as the* (canonical) piecewise representation affine in the first argument *induced by* $\hat{\gamma}$.

End of Fact 9.

Fact 10. *Let*

$$\gamma = \left\{ \langle P, f_P \rangle \right\}_{P \in \mathcal{P}}$$

be a piecewise representation that is affine in the first argument, and let

$$\hat{\gamma} = \left\{ \langle P, A_P, b_P \rangle \right\}_{P \in \mathcal{P}}$$

be a corresponding coefficient-based representation as in Fact 9.

For every $u \in U$ *and every* $P \in \mathcal{P}$*, define*

$$P_u := \{ x \in X \mid (x, u) \in P \}.$$

Then

$$\mathcal{P}_u := \{P_u \mid P \in \mathcal{P},\ P_u \neq \emptyset\}$$

is a finite partition of X.

Furthermore, defining

$$f_u : X \to Y, \qquad f_u(x) := \langle\!\langle \gamma \rangle\!\rangle(x, u),$$

we have that f_u *is piecewise affine.*

More precisely, for every nonempty $P_u \in \mathcal{P}_u$*,*

$$f_u(x) = A_P \cdot x + b_P(u) \qquad \forall x \in P_u,$$

so that $\langle A_P, b_P(u)\rangle$ *are affine coefficients for* f_u *on the region* P_u. End of Fact 10.

A.10 Analytic Euclidean Maps and Neural Primitives

We first isolate the function class used by the RNC constructions in this appendix. The relevant notion is deliberately stronger than ordinary analyticity on the relative domain of a Euclidean space: we require an analytic map on the entire Euclidean ambient space. This is the condition needed below to obtain uniform value and derivative bounds on arbitrary bounded subsets, without additional assumptions that such subsets remain a positive distance from the boundary of an open domain.

We first recall the ordinary notion of analyticity.

Definition 37 (Ordinarily-analytic Euclidean maps, cf. [50])**.** Let $\Omega_X = \mathbb{R}^m$ and let $X \subseteq \Omega_X$ be open. A Euclidean function $f : X \to \mathbb{R}^n$ is *ordinarily-analytic* if, for every $c \in X$, there exist $r \in \mathbb{R}^{>0}$ and coefficients $c_\alpha \in \mathbb{R}^n$, for $\alpha \in \mathbb{N}^m$, such that, letting $\hat{X} := \mathrm{Ball}_{\Omega_X}(c, r)$, the following three conditions hold jointly:

(1) the inclusion $\hat{X} \subseteq X$ holds;

(2) function f is given by the power series

$$f(x) \;=\; \sum_{\alpha \in \mathbb{N}^m} \left(c_\alpha \cdot \textstyle\prod_{i=1}^{m} \left(x_i - c_i\right)^{\alpha_i} \right) \qquad \forall x \in \hat{X}\,;$$

(3) the power series of the above Point (2) converges absolutely on $\hat{X}$.

End of Definition 37.

Definition 38 (Analytic Euclidean maps)**.** Let X and Y be Euclidean spaces having ambient spaces $\Omega_X = \mathbb{R}^m$ and $\Omega_Y = \mathbb{R}^n$, respectively.

▸ **SubDefinition 38.1**. A Euclidean map $f : X \to Y$ is *ambiently-analytic* if there exists an ordinarily-analytic Euclidean map $\tilde{f} : \Omega_X \to \Omega_Y$ such that $\tilde{f}|_X = f$. Any such $\tilde{f}$ is called an *analytic ambient extension* of f.

We write $\mathcal{F}_{\mathrm{AN}}(X, Y)$ for the class of analytic Euclidean maps from X to Y. We also write $\mathcal{F}_{\mathrm{AN}}$ for the class of all such maps across arbitrary finite-dimensional Euclidean source and target spaces. *End of SD.38.1* ◂

▸ **SubDefinition 38.2**. In addition to X and Y, let Θ be a Euclidean parameter space having ambient space Ω_Θ.

A parametric Euclidean map $f : X \times \Theta \to Y$ is a *ambiently-analytic* if it admits an ordinarily-analytic ambient extension $\tilde{f} : \Omega_X \times \Omega_\Theta \to \Omega_Y$ such that $\tilde{f}|_{X\times\Theta} = f$.

We write $\mathcal{F}^{\mathrm{P}}_{\mathrm{AN}}(X, Y; \Theta)$ for the class of analytic parametric Euclidean maps from X to Y with parameter space Θ. *End of SD.38.2* ◂

When we say *analytic* we mean *ambiently-analytic* unless specified otherwise. *End of Definition 38.*

Remark 3. The ambient-extension requirement is essential for boundedness arguments used in our results—e.g., Proposition 28. Ordinary analyticity alone would not suffice: for example, $x \mapsto 1/x$ is ordinarily-analytic on $(0,1)$, but maps the bounded set $(0,1)$ to an unbounded set. By contrast, an analytic ambient extension $\tilde{f} : \Omega_Z \to \Omega_Y$ is defined on the entire ambient Euclidean space $\Omega_Z = \mathbb{R}^d$ and it is continuous with continuous derivatives of every order; hence $\tilde{f}$ is bounded on every bounded subset of its domain Ω_Z, and, for every $n \in \mathbb{N}^{>0}$, the same holds for its n-th derivative $\mathsf{D}^n \tilde{f}$. *End of Remark 3.*

Fact 11 (Analytic Euclidean maps are smooth up to the closure)**.** *Every analytic Euclidean map $f : X \to Y$ admits an infinitely-continuously-differentiable ambient extension $\tilde{f} : \Omega_X \to \Omega_Y$. In particular, $\tilde{f}|_{\mathrm{CL}(X)}$ is a continuously-differentiable extension of f to the closure of its domain.*

End of Fact 11.

Proof. By Definition 38, there exists an ordinarily-analytic ambient extension $\tilde{f} : \Omega_X \to \Omega_Y$ such that $\tilde{f}|_X = f$. Since every ordinarily-analytic Euclidean map between finite-dimensional Euclidean spaces is infinitely-continuously-differentiable, $\tilde{f}$ is continuously differentiable on Ω_X. Therefore, $\tilde{f}|_{\mathrm{CL}(X)}$ is a continuously-differentiable extension of f to the closure of its domain. □

Proposition 1 (Analytic maps are closed under tupling and composition)**.** *The class of analytic Euclidean maps is closed under finite function tupling and function composition. Consequently, if every primitive in a class $\mathcal{P}$ is analytic, then the closure $\mathcal{F}$ of $\mathcal{P}$ satisfies*

$$\mathcal{F} \subseteq \mathcal{F}_{\mathrm{AN}}.$$

Proof. For each $i \in [1..k]$, let $f_i : X \to Y_i$ be analytic Euclidean maps with analytic ambient extensions $\tilde{f}_i$. Then

$$(\tilde{f}_1, \ldots, \tilde{f}_k)$$

is ordinarily-analytic on Ω_X and restricts to $(f_1, \ldots, f_k)$ on X. Hence analytic Euclidean maps are closed under finite tupling.

Similarly, let $f : X \to Y$ and $g : Y \to Z$ be analytic Euclidean maps with analytic ambient extensions $\tilde{f} : \Omega_X \to \Omega_Y$ and $\tilde{g} : \Omega_Y \to \Omega_Z$. Then

$$\tilde{g} \circ \tilde{f} : \Omega_X \to \Omega_Z$$

is ordinarily-analytic and restricts to $g \circ f$ on X. Thus analytic Euclidean maps are also closed under composition.

Therefore the class $\mathcal{F}_{\mathrm{AN}}$ contains $\mathcal{P}$ and it is closed under the two operations in Definition 11. By minimality of the closure of $\mathcal{P}$, we conclude $\mathcal{F} \subseteq \mathcal{F}_{\mathrm{AN}}$. □

Definition 39 (Coordinate maps)**.** Let $n, m \in \mathbb{N}^{>0}$ and let $\iota : [1..m] \to [1..n]$.

▸ **SubDefinition 39.1**. The *coordinate map induced by* ι is the Euclidean function

$$\pi_\iota : \mathbb{R}^n \to \mathbb{R}^m \qquad \pi_\iota(x) := \Big((x)_{\iota(1)}, \ldots, (x)_{\iota(m)} \Big).$$

End of SD.39.1 ◂

▸ **SubDefinition 39.2**. We have the following special cases.

A *coordinate projection* is a coordinate map π_ι where ι is strictly increasing.

A *coordinate permutation* is a coordinate map π_ι where ι is a bijection. *End of SD.39.2* ◂

End of Definition 39.

Fact 12 (Coordinate maps are analytic)**.** *Every coordinate map is analytic. In particular, every coordinate projection and every coordinate permutation is analytic.* End of Fact 12.

Proof. Every coordinate map is a linear, and hence polynomial, map on its full Euclidean ambient space. It is therefore ordinarily-analytic on its full Euclidean ambient space, and consequently ambiently-analytic. □

Proposition 2 (Analytic neural primitives)**.** *The following "neural primitives" are ambiently-analytic, as they are ordinarily-analytic on their full Euclidean ambient spaces.*

- *Analyticity is satisfied by every affine map $x \mapsto A \cdot x + b$ and every parametric affine map $(x, A, b) \mapsto A \cdot x + b$; and in particular by every coordinate projection and permutation.*
- *For every $d \in \mathbb{N}^{>0}$ and every $\varepsilon > 0$, analyticity is satisfied by the parametric LayerNorm map*

$$\mathrm{LN}_\varepsilon : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d\,, \qquad \mathrm{LN}_\varepsilon(x, \gamma, \beta) \;:=\; \gamma * \frac{x - \mu(x) \cdot \mathbf{1}_d}{\sqrt{v(x) + \varepsilon}} + \beta\,,$$

where

$$\mu(x) \;:=\; \frac{1}{d} \cdot \sum_{i=1}^{d} x_i\,, \qquad v(x) \;:=\; \frac{1}{d} \cdot \sum_{i=1}^{d} \big(x_i - \mu(x)\big)^2\,,$$

with $*$ denoting elementwise product.

- *For every $d \in \mathbb{N}^{>0}$ and every $\varepsilon > 0$, analyticity is satisfied by the parametric RMS-normalisation map*

$$\mathrm{RMSNorm}_\varepsilon : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d\,, \qquad \mathrm{RMSNorm}_\varepsilon(x, \gamma) \;:=\; \gamma * \frac{x}{\sqrt{\big(\|x\|^2/d\big) + \varepsilon}}\,,$$

where $*$ is elementwise product.

- *The scalar 'activation' functions*

$$\mathrm{id}, \qquad \exp, \qquad \sin, \qquad \cos, \qquad \tanh, \qquad \sigma(x) := \frac{1}{1 + e^{-x}},$$

$$\mathrm{softplus}(x) := \log(1 + e^x), \qquad \mathrm{SiLU}(x) := x \cdot \sigma(x), \qquad \mathrm{GELU}(x) := \frac{x}{2} \cdot \left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

are ambiently-analytic on $\mathbb{R}$.

- *The softmax map*

$$\mathrm{softmax}(x)_i := \frac{e^{(x)_i}}{\sum_{j=1}^{d} e^{(x)_j}}$$

is ambiently-analytic on $\mathbb{R}^d$.

Proof. Every affine map is polynomial and is therefore ordinarily-analytic on its full Euclidean ambient space. The same applies to every parametric affine map $(x, A, b) \mapsto A \cdot x + b$, since each of its coordinates is a polynomial in the entries of x, A, and b. In particular, coordinate projections and coordinate permutations are linear maps and are therefore ordinarily-analytic on their full Euclidean ambient spaces.

For LayerNorm, the maps μ and v are polynomial. Since

$$v(x) + \varepsilon \geq \varepsilon > 0,$$

the scalar map $z \mapsto z^{-1/2}$ is evaluated only on the positive half-line, where it is ordinarily-analytic. Closure of ordinary analyticity under composition, multiplication, and addition therefore shows that LN_ε is ordinarily-analytic jointly in x, γ, β on its full Euclidean ambient space.

The RMS-normalisation claim follows analogously from

$$\frac{\|x\|^2}{d} + \varepsilon \geq \varepsilon > 0.$$

The functions id, $\exp$, $\sin$, $\cos$, and $\tanh$ are ordinarily-analytic on $\mathbb{R}$. The sigmoid σ is ordinarily-analytic because $1 + e^{-x} > 0$ for every $x \in \mathbb{R}$. Softplus is ordinarily-analytic because $1 + e^x > 0$ for every $x \in \mathbb{R}$, and the logarithm is ordinarily-analytic on the positive half-line. SiLU is a product

of ordinarily-analytic functions. The error function is ordinarily-analytic on $\mathbb{R}$, and therefore GELU is ordinarily-analytic by composition, multiplication, and addition.

Finally, for every $x \in \mathbb{R}^d$,

$$\sum_{j=1}^{d} e^{(x)_j} > 0.$$

Hence every coordinate of the softmax map is a quotient of ordinarily-analytic functions with a nowhere-vanishing denominator. Therefore, softmax is ordinarily-analytic on $\mathbb{R}^d$.

Thus, each of the preceding maps is ordinarily-analytic on its full Euclidean ambient space, and consequently ambiently-analytic by Definition 38. □

Corollary 2 (Closures of analytic neural primitives)**.** *Let $\mathcal{P}$ be any class of analytic Euclidean functions, including any of the "neural primitives" listed in Proposition 2. Then, every function in the closure of $\mathcal{P}$ is analytic.* End of Corollary 2.

Remark 4 (Parametric maps). The preceding statements apply directly to parametric neural maps by treating parameters as additional Euclidean input coordinates. Thus, for example, affine weights and biases, LayerNorm scale and bias, and RMSNorm scale may all vary as parameters while preserving joint analyticity in the ordinary inputs and parameters.

A.11 Affine Hyperplanes and Semilinear Sets

Definition 40 (Affine hyperplane)**.** Let $d \in \mathbb{N}^{>0}$. For $a \in \mathbb{R}^d \setminus \{0\}$ and $b \in \mathbb{R}$, we define

$$H(a, b) := \left\{ z \in \mathbb{R}^d \;\middle|\; a^\top \cdot z = b \right\}.$$

We call every set of this form an *affine hyperplane* in $\mathbb{R}^d$. *End of Definition 40.*

Definition 41 (Affine equalities and inequalities)**.** Let $d \in \mathbb{N}^{>0}$, and let

$$h : \mathbb{R}^d \to \mathbb{R}$$

be an affine function.

An expression of the form

$$h(z) = 0$$

is called an *affine equality*.

An expression of the form

$$h(z) \geq 0$$

is called an *affine inequality*, and an expression of the form

$$h(z) > 0$$

is called a *strict affine inequality*. *End of Definition 41.*

Remark 5. Affine inequalities of the forms

$$h(z) \leq 0 \qquad \text{and} \qquad h(z) < 0$$

are included in the preceding terminology, since they are equivalently written as

$$-h(z) \geq 0 \qquad \text{and} \qquad -h(z) > 0,$$

respectively. *End of Remark 5.*

Definition 42 (Finite affine constraint system)**.** Let $d \in \mathbb{N}^{>0}$. A *finite affine constraint system* on $\mathbb{R}^d$ is a finite collection of affine equalities, affine inequalities, and strict affine inequalities.

Equivalently, it may be represented by finite index sets I, J, K and affine functions

$$h_i, g_j, q_k : \mathbb{R}^d \to \mathbb{R}, \qquad i \in I, \quad j \in J, \quad k \in K,$$

through the system

$$\begin{aligned} h_i(z) &= 0 && \forall i \in I, \\ g_j(z) &\geq 0 && \forall j \in J, \\ q_k(z) &> 0 && \forall k \in K. \end{aligned}$$

End of Definition 42.

Definition 43 (Affine-constraint set)**.** Let $d \in \mathbb{N}^{>0}$. An *affine-constraint set* in $\mathbb{R}^d$ is the solution set of a finite affine constraint system.

Equivalently, it is a set of the form

$$P = \Big\{ z \in \mathbb{R}^d \;\Big|\; \textstyle\bigwedge_{i \in I} h_i(z) = 0\,,\ \bigwedge_{j \in J} g_j(z) \geq 0\,,\ \bigwedge_{k \in K} q_k(z) > 0 \Big\},$$

where I, J, K are finite index sets and

$$h_i, g_j, q_k : \mathbb{R}^d \to \mathbb{R}$$

are affine functions. *End of Definition 43.*

Definition 44 (Semilinear set)**.** Let $d \in \mathbb{N}^{>0}$. A set $S \subseteq \mathbb{R}^d$ is *semilinear* if it is a finite union of affine-constraint sets.

Equivalently, there exist $m \in \mathbb{N}$ and affine-constraint sets

$$P_1, \ldots, P_m \subseteq \mathbb{R}^d$$

such that

$$S = \bigcup_{\ell=1}^{m} P_\ell.$$

End of Definition 44.

Definition 45 (Coordinate-equality and coordinate-threshold hyperplanes)**.** Let $n \in \mathbb{N}^{>0}$.

For $j, k \in [1..n]$ with $j < k$, we define the *coordinate-equality hyperplane*

$$H^{\mathrm{eq}}_{j,k} := \big\{ x \in \mathbb{R}^n \;\big|\; (x)_j = (x)_k \big\}.$$

For $j \in [1..n]$ and $c \in \mathbb{R}$, we define the *coordinate-threshold hyperplane*

$$H^{\mathrm{th}}_{j}(c) := \big\{ x \in \mathbb{R}^n \;\big|\; (x)_j = c \big\}.$$

End of Definition 45.

Definition 46 (Cells induced by a finite hyperplane family)**.** Let $d \in \mathbb{N}^{>0}$, and let $\mathcal{H}$ be a finite family of affine hyperplanes in $\mathbb{R}^d$.

For an affine hyperplane

$$H = H(a, b)\,,$$

we define the relation $\equiv_H$ on $\mathbb{R}^d$ by

$$\begin{aligned} z \equiv_H z' \quad \Longleftrightarrow \quad & \big(a^\top \cdot z - b\big)\big(a^\top \cdot z' - b\big) > 0 \\ & \text{or} \quad \big(a^\top \cdot z - b\big) = \big(a^\top \cdot z' - b\big) = 0. \end{aligned}$$

Thus, $z \equiv_H z'$ if and only if z and z' lie on the same side of H, or both lie on H.

We define the relation $\equiv_{\mathcal{H}}$ on $\mathbb{R}^d$ by

$$z \equiv_{\mathcal{H}} z' \quad \Longleftrightarrow \quad z \equiv_H z' \qquad \forall H \in \mathcal{H}.$$

The *hyperplane cells induced by* $\mathcal{H}$ are the equivalence classes of $\equiv_{\mathcal{H}}$. We denote the corresponding partition of $\mathbb{R}^d$ by

$$\mathcal{C}(\mathcal{H}) := \mathbb{R}^d / \equiv_{\mathcal{H}}\,.$$

End of Definition 46.

Fact 13 (Finiteness of hyperplane-cell partitions)**.** *Let $d \in \mathbb{N}^{>0}$, and let $\mathcal{H}$ be a finite family of affine hyperplanes in $\mathbb{R}^d$. Then*

$$\mathcal{C}(\mathcal{H})$$

is a finite partition of $\mathbb{R}^d$. In particular,

$$\big|\mathcal{C}(\mathcal{H})\big| \;\leq\; 3^{|\mathcal{H}|}.$$

End of Fact 13.

Proof. By Definition 46, every hyperplane $H \in \mathcal{H}$ partitions $\mathbb{R}^d$ into three equivalence classes under $\equiv_H$: the hyperplane H itself and its two open sides.

Hence, every equivalence class of $\equiv_{\mathcal{H}}$ is determined by specifying, for every $H \in \mathcal{H}$, which of these three classes contains its elements. Therefore there are at most

$$3^{|\mathcal{H}|}$$

equivalence classes.

Since $\mathcal{C}(\mathcal{H})$ is, by definition, the set of equivalence classes of $\equiv_{\mathcal{H}}$, it is a finite partition of $\mathbb{R}^d$, and

$$\left|\mathcal{C}(\mathcal{H})\right| \leq 3^{|\mathcal{H}|}.$$

□

A.12 PAP Functions and Intensional Derivatives

We introduce the notions of *piecewise-analytic-under-analytic-partition (PAP)* functions and *intensional derivatives* introduced by [43]. We restate only the definitions and results needed below; see Definitions 2–7 and Propositions 6–10 of [43] for the original development.

Definition 47 (PAP functions)**.** Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$.

Following Definition 3 of [43], a set $A \subseteq \mathbb{R}^n$ is *analytic* if it can be described by finitely many strict and non-strict inequalities of real-analytic functions on open domains; that is, if there exist finite families of real-analytic functions

$$g_j^+ : U_j^+ \to \mathbb{R}, \qquad g_\ell^- : U_\ell^- \to \mathbb{R},$$

with open domains $U_j^+, U_\ell^- \subseteq \mathbb{R}^n$, such that

$$A = \left\{ x \in \mathbb{R}^n \;\middle|\; \bigwedge_j \big(x \in U_j^+ \wedge g_j^+(x) > 0\big) \wedge \bigwedge_\ell \big(x \in U_\ell^- \wedge g_\ell^-(x) \leq 0\big) \right\}.$$

A partition is called *analytic* if all of its sets are analytic.

A *piecewise representation* of a function $f : X \to Y$ is a finite or countable family

$$\gamma = \big\{(A_i, f_i)\big\}_{i \in I},$$

where $\{A_i\}_{i \in I}$ is a partition of X and, for every $i \in I$,

$$f_i : U_i \to Y$$

is defined on an open set $U_i \subseteq \mathbb{R}^n$ containing A_i. Its evaluation is the function $[\![\gamma]\!] : X \to Y$ given by

$$[\![\gamma]\!](x) = f_i(x) \qquad \text{for the unique } i \in I \text{ such that } x \in A_i.$$

The representation γ is a *piecewise-analytic-under-analytic-partition representation*, abbreviated as a *PAP representation*, if $\{A_i\}_{i \in I}$ is an analytic partition and every f_i is real-analytic.

A function $f : X \to Y$ is a *PAP function* if it is the evaluation of some PAP representation:

$$f = [\![\gamma]\!].$$

We write

$$\mathcal{F}_{\mathrm{PAP}}(X, Y)$$

for the class of PAP functions from X to Y. *End of Definition 47.*

Definition 48 (Intensional derivative)**.** Let

$$f : X \to Y, \qquad X \subseteq \mathbb{R}^n, \qquad Y \subseteq \mathbb{R}^m,$$

be a PAP function, and let

$$\gamma = \big\{(A_i, f_i)\big\}_{i \in I}$$

be a PAP representation of f.

The *intensional derivative of the representation* γ is the representation

$$D\gamma := \big\{(A_i, Df_i)\big\}_{i\in I}.$$

By Proposition 6 of [43], $D\gamma$ is itself a PAP representation.

A function

$$df : X \to \mathbb{R}^{m\times n}$$

is an *intensional derivative of* f if there exists a PAP representation γ of f such that

$$df = [\![D\gamma]\!].$$

We write

$$\partial_\bullet f$$

for the set of all intensional derivatives of f. *End of Definition 48.*

Proposition 3 (Almost-everywhere correctness of intensional derivatives)**.** *Let*

$$f : X \to Y, \qquad X \subseteq \mathbb{R}^n, \qquad Y \subseteq \mathbb{R}^m,$$

be a PAP function, and let

$$df \in \partial_\bullet f$$

be an intensional derivative of f*. Then there exists a full-measure subset*

$$\hat{X} \subseteq X$$

such that f *is differentiable at every* $x \in \hat{X}$ *and*

$$Df(x) = df(x) \qquad \forall x \in \hat{X}.$$

Proof. This is the first-order case of Proposition 8 of Lee et al. [43]. □

Proposition 4 (Composition and chain rule for intensional derivatives)**.** *Let*

$$f : X \to Y, \qquad g : Y \to Z,$$

where

$$X \subseteq \mathbb{R}^n, \qquad Y \subseteq \mathbb{R}^m, \qquad Z \subseteq \mathbb{R}^\ell,$$

and suppose that f *and* g *are PAP functions. Then:*

(1) the composition $g \circ f$ *is a PAP function;*

(2) for every

$$df \in \partial_\bullet f \qquad \text{and} \qquad dg \in \partial_\bullet g,$$

the function

$$dh(x) := dg\big(f(x)\big) \cdot df(x)$$

is an intensional derivative of $h := g \circ f$*, that is,*

$$dh \in \partial_\bullet(g \circ f).$$

Proof. The first claim is Proposition 9 of [43], and the second claim is Proposition 10 of [43]. □

Lemma 1 (Finite tupling of intensional derivatives)**.** *For* $i \in [1..k]$*, let*

$$f_i : X \to \mathbb{R}^{m_i}$$

be PAP functions having intensional derivatives

$$df_i \in \partial_\bullet f_i.$$

Define

$$f := (f_1, \ldots, f_k)$$

and

$$df(x) := \begin{pmatrix} df_1(x) \\ \vdots \\ df_k(x) \end{pmatrix}.$$

Then f *is PAP and*

$$df \in \partial_\bullet f.$$

Proof. For every $i \in [1..k]$, since df_i is an intensional derivative of f_i, there exists a PAP representation

$$\gamma_i = \{(A_{i,j}, f_{i,j})\}_{j \in I_i}$$

of f_i such that

$$df_i = [\![D\gamma_i]\!].$$

Consider the finite common refinement of these representations. For every

$$\alpha = (j_1, \ldots, j_k) \in I_1 \times \cdots \times I_k,$$

let

$$A_\alpha := \bigcap_{i=1}^{k} A_{i,j_i},$$

and define

$$f_\alpha := (f_{1,j_1}, \ldots, f_{k,j_k})$$

on the intersection of the corresponding open domains.

Finite intersections of analytic sets are analytic, finite intersections of open sets are open, and finite tuples of analytic functions are analytic. Furthermore, $I_1 \times \cdots \times I_k$ is countable. Hence

$$\gamma := \{(A_\alpha, f_\alpha)\}_{\alpha \in I_1 \times \cdots \times I_k}$$

is a PAP representation of f. This is the finite-tupling closure of PAP functions also used in the proof of Proposition 11 of [43].

Furthermore, for every α,

$$Df_\alpha(x) = \begin{pmatrix} Df_{1,j_1}(x) \\ \vdots \\ Df_{k,j_k}(x) \end{pmatrix}.$$

It follows that

$$[\![D\gamma]\!](x) = \begin{pmatrix} df_1(x) \\ \vdots \\ df_k(x) \end{pmatrix} = df(x).$$

Hence $df \in \partial_\bullet f$. □

Remark 6. Proposition 7 of [43] shows that every PAP function admits intensional derivatives of every order. Proposition 8 shows that every first-order intensional derivative agrees with the ordinary derivative wherever required outside a measure-zero set. These results motivate the notion of intensional derivative but are not needed for Proposition 24; its proof uses only the definitions above, finite tupling, and the composition and chain-rule results of Propositions 9 and 10 of [43].

A.13 Lattice Theory

We present the elementary lattice-theoretic notions of *meet*, *join*, *lattice*, and *distributive lattice* used in this work—cf. [51].

Definition 49. A *partially ordered set* (or *poset*) is a pair $(X, \preceq)$ where $\preceq$ is a reflexive, antisymmetric, and transitive binary relation on X. *End of Definition 49.*

Definition 50. Let $(X, \preceq)$ be a poset, and let $x, y \in X$.

An element $z \in X$ is:

- a *lower bound* of x and y if $z \preceq x$ and $z \preceq y$;
- an *upper bound* of x and y if $x \preceq z$ and $y \preceq z$.

The *meet* of x and y, when it exists, is their greatest lower bound and is denoted by $x \wedge y$. Thus

$$x \wedge y \preceq x, \qquad x \wedge y \preceq y,$$

and, for every $z \in X$,

$$z \preceq x \ \wedge \ z \preceq y \quad \Longrightarrow \quad z \preceq x \wedge y.$$

Dually, the *join* of x and y, when it exists, is their least upper bound and is denoted by $x \vee y$. Thus

$$x \preceq x \vee y, \qquad y \preceq x \vee y,$$

and, for every $z \in X$,

$$x \preceq z \ \wedge \ y \preceq z \quad \Longrightarrow \quad x \vee y \preceq z.$$

Whenever they exist, meets and joins are unique by antisymmetry. *End of Definition 50.*

Definition 51 (Lattice)**.** A poset $(X, \preceq)$ is a *lattice* if every pair $x, y \in X$ admits both a meet $x \wedge y$ and a join $x \vee y$. *End of Definition 51.*

Fact 14. *Meet and join are associative in every lattice. Hence finite non-empty meets and joins are unambiguously denoted by*

$$\bigwedge_{i=1}^{n} x_i \qquad \textit{and} \qquad \bigvee_{i=1}^{n} x_i,$$

respectively. End of Fact 14.

Definition 52 (Distributive lattice)**.** A lattice $(X, \preceq)$ is *distributive* if, for every $x, y, z \in X$,

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z).$$

End of Definition 52.

Fact 15. *In a distributive lattice, the binary distributive laws extend to arbitrary finite non-empty meets and joins. In particular, for every $n \in \mathbb{N}^{>0}$,*

$$x \wedge \bigvee_{i=1}^{n} y_i \ = \ \bigvee_{i=1}^{n} (x \wedge y_i), \qquad\qquad x \vee \bigwedge_{i=1}^{n} y_i \ = \ \bigwedge_{i=1}^{n} (x \vee y_i).$$

End of Fact 15.

A.14 Algebraic Structures

We recall the elementary notions of universal algebra used in this work; cf. [52]. Here, the term *algebra* is used in the universal-algebraic sense of a set equipped with a collection of finitary operations, rather than in the more specialised sense of an algebra over a field.

Definition 53 (Finitary operation)**.** Let X be a non-empty set, and let $n \in \mathbb{N}$.

An *n-ary operation on X* is a function $f : X^n \to X$. The integer n is called the *arity* of f.

An operation of finite arity is called a *finitary operation*. In particular, operations of arity 1 and 2 are called *unary* and *binary*, respectively. *End of Definition 53.*

Definition 54 (Algebraic signature)**.** An *algebraic signature* is a pair $\Sigma = (F, \mathrm{ar}())$ where F is a set of operation symbols and $\mathrm{ar}(:)F \to \mathbb{N}$ assigns an *arity* to every operation symbol.

When the arity of every operation symbol is clear from context, we identify the signature with F. *End of Definition 54.*

Definition 55 (Algebra)**.** Let $\Sigma = (F, \mathrm{ar}())$ be an algebraic signature.

A *Σ-algebra* is a tuple $\mathcal{A} = (A, (f^{\mathcal{A}})_{f \in F})$ where:

- A is a non-empty set, called the *carrier* or *underlying set* of $\mathcal{A}$;
- for every $f \in F$, $f^{\mathcal{A}} : A^{\mathrm{ar}(()f)} \to A$ is an operation on A.

When no ambiguity arises, we omit the superscript $\mathcal{A}$ from the operations. *End of Definition 55.*

Definition 56 (Binary algebra)**.** A *binary algebra* $\mathcal{A}$ is an algebra whose signature consists only of binary operation symbols.

In particular, an algebra having two binary operations $\star$ and $\diamond$ is written as

$$\mathcal{A} = (A, \star, \diamond) \quad \text{where} \quad \star, \diamond : A \times A \to A.$$

Binary operations are written using *infix notation*. Given a binary operation $\star : A \times A \to A$, and elements $a, b \in A$, we write $a \star b$ in place of $\star(a, b)$. *End of Definition 56.*

Definition 57 (Properties of binary operations)**.** A binary operation $\star$ on a set A is:

- *associative* if
$$(x \star y) \star z \;=\; x \star (y \star z) \qquad \forall x, y, z \in A;$$
- *commutative* if
$$x \star y \;=\; y \star x \qquad \forall x, y \in A;$$
- *idempotent* if
$$x \star x \;=\; x \qquad \forall x \in A.$$

End of Definition 57.

Definition 58 (Distributive binary operations)**.** Let $\star$ and $\diamond$ be binary operations on a set A. Then:

- operation $\diamond$ is *left-distributive over* $\star$ if
$$x \diamond (y \star z) = (x \diamond y) \star (x \diamond z) \qquad \forall x, y, z \in A;$$
- operation $\diamond$ is *right-distributive over* $\star$ if
$$(x \star y) \diamond z = (x \diamond z) \star (y \diamond z) \qquad \forall x, y, z \in A;$$
- operation $\diamond$ *distributes over* $\star$ if it is both left-distributive and right-distributive over $\star$;
- operations $\star$ and $\diamond$ are *mutually distributive* if $\diamond$ distributes over $\star$ and $\star$ distributes over $\diamond$.

End of Definition 58.

Definition 59 (Subalgebra)**.** Let $\mathcal{A} = (A, (f^{\mathcal{A}})_{f \in F})$ be a Σ-algebra.

A *subalgebra* of $\mathcal{A}$ is a Σ-algebra $\mathcal{B} = (B, (f^{\mathcal{B}})_{f \in F})$ such that

$$B \subseteq A \qquad \text{and} \qquad f^{\mathcal{B}} = f^{\mathcal{A}}\big|_{B^{\mathrm{ar}(()f)}} \quad \forall f \in F.$$

Equivalently, B is closed under all operations of $\mathcal{A}$, and $\mathcal{B}$ carries their restrictions to B. *End of Definition 59.*

Definition 60 (Homomorphism and isomorphism)**.** Let us consider Σ-algebras

$$\mathcal{A} = (A, (f^{\mathcal{A}})_{f \in F}) \qquad \text{and} \qquad \mathcal{B} = (B, (f^{\mathcal{B}})_{f \in F}).$$

A function $h : A \to B$ is a *homomorphism* if, for every operation symbol $f \in F$ of arity $n = \mathrm{ar}(()f)$,

$$h\big(f^{\mathcal{A}}(a_1, \ldots, a_n)\big) = f^{\mathcal{B}}\big(h(a_1), \ldots, h(a_n)\big) \qquad \forall\, (a_1, \ldots, a_n) \in A^n.$$

A bijective homomorphism is called an *isomorphism*. Two algebras are *isomorphic* if there exists an isomorphism between them. *End of Definition 60.*

Definition 61 (Lattice algebra)**.** A *lattice algebra* is a binary algebra

$$\mathcal{A} = (A, \vee, \wedge)$$

whose two operations are associative, commutative, and idempotent, and satisfy the following identities for every $x, y \in A$:

$$x \vee (x \wedge y) \;=\; x,$$
$$x \wedge (x \vee y) \;=\; x.$$

A lattice algebra is *distributive* if $\vee$ and $\wedge$ are mutually distributive. *End of Definition 61.*

Fact 16 (Order and algebraic presentations of lattices)**.** *The order-theoretic and algebraic notions of lattice are equivalent.*

In particular, if $(A, \preceq)$ *is a lattice in the sense of Definition 51, then*

$$(A, \vee, \wedge)$$

is a lattice algebra, where $\vee$ *and* $\wedge$ *are respectively the join and meet induced by* $\preceq$*.*

Conversely, a lattice algebra $(A, \vee, \wedge)$ *induces the partial order*

$$x \preceq y \quad \Longleftrightarrow \quad x \vee y = y,$$

equivalently,

$$x \preceq y \quad \Longleftrightarrow \quad x \wedge y = x.$$

Under these correspondences, distributive lattices correspond exactly to distributive lattice algebras.
End of Fact 16.

A.15 Semigroup and Group Theory

We recall the elementary notions of semigroup, group, and permutation-group theory used in this work; cf. [53] for semigroup theory and [54] for permutation groups. Semigroups and groups are particular algebraic structures in the sense of Section A.14.

Definition 62 (Semigroup)**.** A *semigroup* is a binary algebra

$$S = (A_S, \cdot_S)$$

whose binary operation

$$\cdot_S : A_S \times A_S \to A_S$$

is associative:

$$(a \cdot_S b) \cdot_S c = a \cdot_S (b \cdot_S c) \qquad \forall a, b, c \in A_S.$$

The semigroup S is *finite* if its carrier A_S is finite. *End of Definition 62.*

Definition 63 (Products of subsets of a semigroup)**.** Let

$$S = (A_S, \cdot_S)$$

be a semigroup, and let $B, C \subseteq A_S$.

Their product is

$$B \cdot_S C := \{b \cdot_S c \mid b \in B,\, c \in C\}.$$

For a non-empty subset $B \subseteq A_S$, its positive powers are defined recursively by

$$B^1 := B,$$
$$B^{n+1} := B^n \cdot_S B \qquad \forall n \in \mathbb{N}^{>0},$$

and

$$B^+ := \bigcup_{n \in \mathbb{N}^{>0}} B^n.$$

End of Definition 63.

Fact 17. *Let* S *be a semigroup and let* B *be a non-empty subset of its carrier. Then* B^+ *is the smallest subsemigroup of* S *whose carrier contains* B*. It is therefore the* subsemigroup generated by B*.*
End of Fact 17.

Definition 64 (Group)**.** A *group* is a semigroup

$$G = (A_G, \cdot_G)$$

satisfying the following properties:

(1) there exists an element $e \in A_G$ such that

$$e \cdot_G a = a \cdot_G e = a \qquad \forall a \in A_G;$$

this element is unique and is called the *identity element* of G;

(2) for every $a \in A_G$, there exists an element $b \in A_G$ such that

$$a \cdot_G b = b \cdot_G a = e;$$

this element is unique and is called the *inverse* of a, denoted by a^{-1}.

The group G is *finite* if its carrier A_G is finite. *End of Definition 64.*

Definition 65 (Subsemigroup and subgroup)**.** Let

$$S = (A_S, \cdot_S)$$

be a semigroup.

A *subsemigroup* of S is a subalgebra

$$S' = (A_{S'}, \cdot_{S'})$$

of S. Equivalently, $A_{S'}$ is a non-empty subset of A_S closed under $\cdot_S$, and $\cdot_{S'}$ is the restriction of $\cdot_S$ to $A_{S'}$.

Let G be a group. A *subgroup* of G is a subsemigroup of G that is itself a group. *End of Definition 65.*

Fact 18. *Homomorphisms and isomorphisms between semigroups or groups are precisely homomorphisms and isomorphisms in the sense of Definition 60 for their corresponding algebraic signatures.*

In particular, an isomorphism between semigroups

$$S_1 = (A_1, \cdot_1) \qquad \textit{and} \qquad S_2 = (A_2, \cdot_2)$$

is a bijection

$$\psi : A_1 \to A_2$$

satisfying

$$\psi(a \cdot_1 b) = \psi(a) \cdot_2 \psi(b) \qquad \forall a, b \in A_1.$$

End of Fact 18.

Definition 66 (Transformation semigroup and group)**.** Let X be a set.

A *transformation semigroup* on X is a pair

$$(X, T)$$

where T is a set of transformations on X, as per Definition 7, such that

$$(T, \circ)$$

is a semigroup under function composition.

A *transformation group* on X is a transformation semigroup (X, T) such that

$$(T, \circ)$$

is a group. *End of Definition 66.*

Definition 67 (Permutation group)**.** A *permutation group* on a set X is a transformation group

$$(X, P)$$

such that every element of P is a permutation of X as per Definition 7.

If X is finite, the *degree* of the permutation group is

$$|X|.$$

End of Definition 67.

Definition 68 (Symmetric group)**.** Let X be a set. The *symmetric group* on X, denoted by $\mathrm{Sym}(X)$, is the permutation group

$$(X, P_X),$$

where P_X is the set of all permutations of X, equipped with function composition.

Every permutation group on X is a subgroup of $\mathrm{Sym}(X)$. *End of Definition 68.*

Theorem 10 (Cayley)**.** *Every group*

$$G = (A_G, \cdot_G)$$

is isomorphic to a subgroup of $\mathrm{Sym}(A_G)$.

A.16 Algebraic Automata Theory

Algebraic Automata Theory (AAT), cf. [35, 36, 55, 56], offers a refined perspective on automata, seeing an automaton not as a transition system (a graph where one moves from a node to another according to a given input), but instead as a device that manipulates elements of an underlying set by applying transformations selected by the inputs. As as result, it connects automata to semigroup and group theory, allowing for advanced characterisations automata.

An *automaton* is a system $A = \langle Q, \Sigma, \delta, q_0, \Gamma, \theta \rangle$ where Q, Σ, Γ are finite spaces under the discrete metric. The standard terminology differs from the one of dynamical systems as Σ is called input alphabet, δ is called transition function, Γ is called output alphabet, and the tuple $D = \langle Q, \Sigma, \delta \rangle$ is called a *semiautomaton*.

For $\sigma \in \Sigma$, the *state transformation* δ_σ of D is the function $\delta_\sigma : Q \to Q$ given by $\delta_\sigma(q) = \delta(q, \sigma)$. Semiautomaton D is an *identity-reset semiautomaton* if each transformation δ_σ is the identity function or a constant function, and it is a *permutation semiautomaton* if each transformation δ_σ is a bijection.

The *transition monoid* of D is the transformation monoid $\mathsf{M}(D) = (X, \circ)$ where

$$X = \{\mathrm{id}\} \cup \{\delta_{\sigma_1} \circ \cdots \circ \delta_{\sigma_n} \mid \sigma_1, \ldots, \sigma_n \in \Sigma\}.$$

For $\mathsf{G} = \langle G, \gamma \rangle$ a finite group with elements G and operator $\gamma : G \times G \to G$, the G-*semiautomaton* is $A_\mathsf{G} = \langle G, G, \gamma \rangle$.

A *Mealy machine* is a tuple $M = \langle Q, \Sigma, \delta, \Gamma, \theta \rangle$ where $D_M = \langle Q, \Sigma, \delta \rangle$ is a semiautomaton, Γ is an output alphabet, and $\theta : Q \times \Sigma \to \Gamma$ is an output function. It defines the mapping $Q \times \Sigma^+ \to \Gamma$ given by

$$M(q, \sigma_1 \cdots \sigma_n) = \theta\big(D_M(q, \sigma_1 \cdots \sigma_n), \sigma_n\big).$$

Given an automaton $A = \langle Q, \Sigma, \delta, q_0, \Gamma, \theta \rangle$, the *associated Mealy machine* $M_A = \langle Q, \Sigma, \delta, \Gamma, \theta \rangle$ is obtained by dropping the initial state from A.

Given a semiautomaton $D_A = \langle Q, \Sigma, \delta \rangle$, its *canonical Mealy machine* is

$$\mathcal{M}(D) := \langle Q, \Sigma, \delta, \Gamma, \theta \rangle, \quad \text{where } \Gamma := Q \times \Sigma, \text{ and } \theta := \mathrm{id}.$$

Definition 69 (Assignments and realisations; Definitions 1.14 and 1.15 of [36])**.** If $M = \langle Q, \Sigma, \delta, \Gamma, \theta \rangle$ and $M' = \langle Q', \Sigma', \delta', \Gamma', \theta' \rangle$ are Mealy Machines, then the triple (α, ι, ζ) is called an *assignment* of M into M' when the functions

$$\alpha : Q \to \mathcal{P}_+(Q'), \quad \iota : \Sigma \to \Sigma', \quad \zeta : \Gamma' \to \Gamma,$$

satisfy the two conditions below for every $q \in Q$, every $q' \in \alpha(q)$, and every $\sigma \in \Sigma$.

$$\text{I)} \quad \delta'\big(q', \iota(\sigma)\big) \in \alpha\big(\delta(q, \sigma)\big)$$
$$\text{II)} \quad \zeta \circ \theta'\big(q', \iota(\sigma)\big) = \theta\big(q, \sigma\big)$$

If an assignment of M into M' exists, then we say that M' is a *realisation* of M or, equivalently, that M' *realises* M.

Fact 19 (§1.3 of [36])**.** *If M is a realisation of M' and M' is a realisation of M'', then M realies M''.*

Theorem 11 (Theorem 1.5 in §1.3 of [36])**.** *If $M' = \langle Q', \Sigma', \delta', \Gamma', \theta' \rangle$ is a realisation of $M = \langle Q, \Sigma, \delta, \Gamma, \theta \rangle$ through an assignment (α, ι, ζ), then for all $q_0 \in Q$, $\mathbf{w} := w_{[1..\ell]} \in \Sigma^+$, and $q_0' \in \alpha(q_0)$, it holds that*

$$\theta\big(M(q_0, \mathbf{w}), w_\ell\big) = \zeta \circ \theta'\big(M'(q_0', \iota(\mathbf{w})), \iota(w_\ell)\big),$$

i.e., $M(q_0, \mathbf{w}) = \zeta \circ M'\big(q_0', \iota(\mathbf{w})\big)$.

Theorem 12 (Krohn-Rhodes, cf. Theorem 7.9 in [36])**.** *Let $A = \langle Q, \Sigma, \delta \rangle$ be a semiautomaton, and let $\mathbf{G}$ be the set of simple groups that are factors of a subgroup of the transition monoid of A. Then, A is realised by a cascade $A_1 \ltimes \cdots \ltimes A_m$ where each A_i is either a two-state identity-reset semiautomaton or a G-semiautomaton for some $\mathsf{G} \in \mathbf{G}$.*

Corollary 3. *Let $A = \langle Q, \Sigma, \delta \rangle$ be a semiautomaton. Then, A is realised by a cascade $A_1 \ltimes \cdots \ltimes A_m$ where each A_i is an identity-reset or a permutation semiautomaton.*

Proof. Implies by the Krohn-Rhodes theorem since the state transformations of any G-semiautomaton $A_{\mathsf{G}} = \langle G, G, \gamma \rangle$ are bijections. In fact, any transformation is given by $\gamma_g(g') = \gamma(g', g)$, which is invertible since every group element admits an inverse element, and hence it is a bijection. □

A.17 Metric Spaces and Ambient Spaces

We follow the standard terminology for metric spaces and topology (cf. 57), consistently with Metric Automata Theory [37]. In particular, spaces are regarded intrinsically as metric spaces. When useful, we additionally regard a metric space as a subspace of a designated *ambient space*; this allows us to refer to structures of the ambient space, such as Euclidean coordinates and norms, without requiring the space itself to coincide with the ambient space.

Definition 70 (Metrics and metric spaces)**.** Let X be a set. A *metric* on X is a function

$$d_X : X \times X \to \mathbb{R}^{\geq 0}$$

such that, for every $x, y, z \in X$,

$$d_X(x, y) = 0 \quad \Longleftrightarrow \quad x = y, \quad d_X(x, y) = d_X(y, x), \quad d_X(x, z) \leq d_X(x, y) + d_X(y, z).$$

A *metric space* is a set equipped with a metric.

We generally identify a metric space with its underlying set and omit its metric from the notation whenever the metric is clear from the context. *End of Definition 70.*

Definition 71 (Metric subspaces and ambient spaces)**.** Let X and Ω_X be metric spaces. We say that X is a *metric subspace* of Ω_X if

$$X \subseteq \Omega_X \qquad \text{and} \qquad d_X = d_{\Omega_X}|_{X \times X}.$$

When Ω_X is a designated metric space of which X is a metric subspace, we call Ω_X the *ambient space* of X. Accordingly, whenever we write

$$X \subseteq \Omega_X,$$

with Ω_X specified as the ambient space of X, the metric on X is understood to be the metric induced by Ω_X. *End of Definition 71.*

Remark 7 (Intrinsic and ambient structure). A metric space need not have a distinguished ambient space, and an ambient realisation need not be unique. Thus, the ambient space is additional structure associated with the presentation of a space rather than part of the abstract notion of metric space.

Metric notions involving only points of X, such as distances and boundedness, are intrinsic to X. When an ambient space Ω_X is specified, additional notions referring to the surrounding space—for example Euclidean coordinates, norms, derivatives, and Lebesgue measure—are understood with respect to Ω_X. *End of Remark 7.*

Definition 72 (Balls and bounded sets)**.** Let X be a metric space. For every $x \in X$ and $r \in \mathbb{R}^{\geq 0}$, the *open ball* and *closed ball* of radius r centred at x are, respectively,

$$\mathrm{Ball}(x, r)X := \big\{ y \in X : d_X(x, y) < r \big\}, \qquad \overline{\mathrm{Ball}}(x, r)X := \big\{ y \in X : d_X(x, y) \leq r \big\}.$$

A set $K \subseteq X$ is *bounded* if there exist $x \in X$ and $r \in \mathbb{R}^{\geq 0}$ such that

$$K \subseteq \overline{B}_X(x, r).$$

End of Definition 72.

Definition 73 (Open and closed sets)**.** Let X be a metric space and let $A \subseteq X$. The set A is *open in* X if, for every $x \in A$, there exists $r > 0$ such that

$$B_X(x, r) \subseteq A.$$

The set A is *closed in* X if $X \setminus A$ is open in X. *End of Definition 73.*

Definition 74 (Interior)**.** Let X be a metric space and let $A \subseteq X$. The *interior of A in X* is the set

$$\operatorname{int}_X(A) := \{x \in A; |; \exists r > 0 : B_X(x, r) \subseteq A\}.$$

If X has ambient space Ω_X, we distinguish the intrinsic interior $\operatorname{int} *X(A)$ from the *ambient interior*

$$\operatorname{int} *\Omega_X(A),$$

namely the interior of A when regarded as a subset of Ω_X. *End of Definition 74.*

Remark 8 (Interior and openness). For every $A \subseteq X$,

$$A \text{ is open in } X \qquad \Longleftrightarrow \qquad \operatorname{int}_X(A) = A.$$

In general, when X is a proper metric subspace of an ambient space Ω_X, one may have

$$\operatorname{int} *X(A) \neq \operatorname{int} *\Omega_X(A).$$

End of Remark 8.

Remark 9 (Subspace and ambient topology). Let Ω_X be the ambient space of X. Openness and closedness of a subset $A \subseteq X$ are understood relative to X unless the ambient space is explicitly mentioned. In particular, A may be open or closed in X without being open or closed in Ω_X.

By contrast, since X carries the metric induced by Ω_X, a subset $A \subseteq X$ is bounded in X if and only if it is bounded as a subset of Ω_X. *End of Remark 9.*

Definition 75 (Product metric spaces)**.** Let $X_1, \dots, X_k$ be metric spaces. Their Cartesian product

$$X_1 \times \cdots \times X_k$$

is equipped with the product metric

$$d\big(\langle x_1, \dots, x_k\rangle, \langle x_1', \dots, x_k'\rangle\big) := \sqrt{\sum_{i=1}^{k} d_{X_i}(x_i, x_i')^2}.$$

If each X_i has ambient space Ω_{X_i}, we take

$$\Omega_{X_1 \times \cdots \times X_k} := \Omega_{X_1} \times \cdots \times \Omega_{X_k}$$

as the canonical ambient space of their product. *End of Definition 75.*

Definition 76 (Euclidean spaces)**.** For every $n \in \mathbb{N}^{>0}$, the *Euclidean n-space* is $\mathbb{R}^n$ equipped with the Euclidean metric

$$d(x, y) := \|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

As usual, we omit the metric and simply write $\mathbb{R}^n$.

A metric space X has a *Euclidean ambient space* if its designated ambient space satisfies

$$\Omega_X = \mathbb{R}^n$$

for some $n \in \mathbb{N}^{>0}$. In this case,

$$X \subseteq \Omega_X = \mathbb{R}^n,$$

and X carries the Euclidean metric restricted to X. *End of Definition 76.*

Remark 10 (Products of Euclidean ambient spaces). Under the canonical identification

$$\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} \cong \mathbb{R}^{n_1 + \cdots + n_k},$$

the product metric above coincides with the Euclidean metric. Hence finite products of spaces having Euclidean ambient spaces again have Euclidean ambient spaces. *End of Remark 10.*

A.18 Measure Theory

We recall the basic notions from measure theory that we use throughout the paper. In particular, whenever a space is contained in a Euclidean ambient space, measurability and almost-everywhere statements are understood with respect to the Lebesgue measure of that ambient space.

We follow the standard terminology for measurable spaces, Lebesgue measure, null sets, almost-everywhere properties, and product measures (cf. 58).

Definition 77 (Measurable spaces and measures)**.** A *σ-algebra* on a set X is a collection $\Sigma \subseteq 2^X$ such that:

- $X \in \Sigma$;
- if $A \in \Sigma$, then $X \setminus A \in \Sigma$;
- if $(A_i)_{i \in \mathbb{N}^{>0}}$ is a sequence of sets in Σ, then

$$\bigcup_{i \in \mathbb{N}^{>0}} A_i \in \Sigma.$$

The pair $\langle X, \Sigma \rangle$ is called a *measurable space*, and the elements of Σ are called *measurable sets*.

A *measure* on $\langle X, \Sigma \rangle$ is a function

$$\mu : \Sigma \to [0, \infty]$$

such that $\mu(\varnothing) = 0$ and, for every sequence of pairwise-disjoint sets $(A_i)_{i \in \mathbb{N}^{>0}}$ in Σ,

$$\mu\left(\bigcup_{i \in \mathbb{N}^{>0}} A_i \right) = \sum_{i \in \mathbb{N}^{>0}} \mu(A_i).$$

The triple $\langle X, \Sigma, \mu \rangle$ is called a *measure space*. *End of Definition 77.*

Definition 78 (Measurable functions)**.** Let $\langle X, \Sigma_X \rangle$ and $\langle Y, \Sigma_Y \rangle$ be measurable spaces. A function

$$f : X \to Y$$

is *measurable* if

$$f^{-1}(A) \in \Sigma_X \qquad \forall A \in \Sigma_Y.$$

End of Definition 78.

Definition 79 (Lebesgue measure)**.** For every $n \in \mathbb{N}^{>0}$, we denote by $\mathcal{L}^n$ the σ-algebra of Lebesgue-measurable subsets of $\mathbb{R}^n$ and by

$$\lambda_n : \mathcal{L}^n \to [0, \infty]$$

the n-dimensional *Lebesgue measure*.

Accordingly,

$$\langle \mathbb{R}^n, \mathcal{L}^n, \lambda_n \rangle$$

is the n-dimensional Lebesgue measure space. *End of Definition 79.*

Definition 80 (Ambient Lebesgue measure)**.** Let X be a metric space having Euclidean ambient space

$$\Omega_X = \mathbb{R}^n,$$

and suppose that X is Lebesgue-measurable in Ω_X.

The *Lebesgue σ-algebra of X* is

$$\mathcal{L}_X := \big\{ A \subseteq X : A \in \mathcal{L}^n \big\},$$

and the *ambient Lebesgue measure on X* is the restriction

$$\lambda_X : \mathcal{L}_X \to [0, \infty], \qquad \lambda_X(A) := \lambda_n(A).$$

When the ambient space is clear from the context, we simply refer to Lebesgue measurability and Lebesgue measure on X. *End of Definition 80.*

Definition 81 (Null sets and full-measure subsets)**.** Let X have a Euclidean ambient space and be Lebesgue-measurable in it.

A measurable set $N \subseteq X$ is *null* or *Lebesgue-null* if

$$\lambda_X(N) = 0.$$

Given measurable sets

$$A \subseteq B \subseteq X,$$

we say that A has *full measure in* B if

$$\lambda_X(B \setminus A) = 0.$$

End of Definition 81.

Definition 82 (Almost-everywhere properties)**.** Let B be a measurable subset of a space having a Euclidean ambient space, and let $\mathcal{P}(x)$ be a property defined for $x \in B$.

We say that $\mathcal{P}$ holds *almost everywhere in* B, abbreviated *a.e. in* B, if the set

$$\big\{x \in B : \mathcal{P}(x) \text{ does not hold}\big\}$$

is Lebesgue-null.

Equivalently, $\mathcal{P}$ holds a.e. in B if there exists a full-measure subset $\hat{B} \subseteq B$ such that $\mathcal{P}(x)$ holds for every $x \in \hat{B}$. *End of Definition 82.*

Fact 20 (Restriction of full-measure subsets)**.** *Let $A \subseteq B \subseteq X$ be measurable sets such that A has full measure in B. Then, for every measurable $C \subseteq B$, the set*

$$A \cap C$$

has full measure in C. End of Fact 20.

Proof. We have

$$C \setminus (A \cap C) = C \setminus A \subseteq B \setminus A.$$

Since $B \setminus A$ is Lebesgue-null, so is $C \setminus (A \cap C)$. □

Fact 21 (Measurability of bounded truncations)**.** *Let X be a Lebesgue-measurable subset of a Euclidean ambient space Ω_X. Then, for every $r \in \mathbb{R}^{\geq 0}$, its bounded truncation*

$$X_{\leq r} = X \cap \overline{B}_{\Omega_X}(0, r)$$

is Lebesgue-measurable and has finite Lebesgue measure. End of Fact 21.

Proof. The closed Euclidean ball $\overline{B}_{\Omega_X}(0, r)$ is Lebesgue-measurable and bounded. Hence its intersection with the measurable set X is measurable. Furthermore,

$$X_{\leq r} \subseteq \overline{B}_{\Omega_X}(0, r),$$

and every bounded Lebesgue-measurable subset of a finite-dimensional Euclidean space has finite Lebesgue measure. □

Fact 22 (Lebesgue measure on product spaces)**.** *Let*

$$\Omega_X = \mathbb{R}^m, \qquad \Omega_Y = \mathbb{R}^n.$$

Under the canonical identification

$$\Omega_X \times \Omega_Y \cong \mathbb{R}^{m+n},$$

the Lebesgue measure on the product ambient space coincides with the product measure

$$\lambda_{m+n} = \lambda_m \times \lambda_n.$$

In particular, if $N \subseteq \Omega_X$ is Lebesgue-null and $A \subseteq \Omega_Y$ is Lebesgue-measurable, then

$$N \times A$$

is Lebesgue-null in $\Omega_X \times \Omega_Y$. End of Fact 22.

Fact 23 (Products preserve full measure)**.** *Let $A \subseteq B \subseteq X$ be measurable sets such that A has full measure in B, and let $C \subseteq Y$ be measurable. Then*

$$A \times C$$

has full measure in

$$B \times C.$$

End of Fact 23.

Proof. We have

$$(B \times C) \setminus (A \times C) = (B \setminus A) \times C.$$

Since $B \setminus A$ is Lebesgue-null, Fact 22 implies that $(B \setminus A) \times C$ is Lebesgue-null. □

Remark 11 (Ambient dimension and almost-everywhere properties). Lebesgue measure is defined with respect to the designated Euclidean ambient space. Consequently, if

$$X \subseteq \Omega_X = \mathbb{R}^n$$

is itself Lebesgue-null in Ω_X, then every measurable subset of X has full measure in X.

For this reason, parameter spaces on which almost-everywhere properties are considered are understood to be represented in their independent Euclidean coordinates, rather than as lower-dimensional subsets of unnecessarily larger ambient spaces. *End of Remark 11.*

A.19 Metric Automata Theory

Metric Automata Theory (MAT) [37] extends the applicability of AAT to arbitrary dynamical systems, with automata as a special case. We introduce the relevant notions.

A *path* in X from a to b is a continuous map $\gamma : [0,1] \to X$ such that $\gamma(0) = a$ and $\gamma(1) = b$. We can define a relation $\sim_X$, where $a \sim_X b$ when there is a path in X from a to b. This relation is an equivalence, partitioning X into equivalence classes $[x]_{\sim_X} = \{x' \in X \mid x' \sim_X x\}$ for $x \in X$, called *(path-connected) components*. For space X, we denote the set of its equivalence classes by $\overline{X}$. Path-connectedness is preserved by every continuous function $f : X \to Y$, and hence the function $\overline{f} : \overline{X} \to \overline{Y}$ with $\overline{f}(\overline{x}) = \overline{f(x)}$ is well-defined.

A metric space X is *η-finite* if it is a finite union of compact, path-connected components. The continuous image of an η-finite space is η-finite (Lemma 18 of [37]). The Cartesian-product $X \times Y$ space of η-finite spaces is η-finite. The η-components of $X \times Y$ are the products η-components of X and η-components of Y (Lemma 19 of [37]).

Lemma 2 (Lemma 22 of [37])**.** *Let X be an η-finite space and Σ a finite alphabet. Then a function $f : X \to \Sigma$ is continuous if and only if it is constant on the η-components of X.*

Dynamics $D = \langle X, U, f \rangle$ and system $S = \langle X, U, f, x_0, Y, h \rangle$ are *η-finite* if X and U are η-finite; in such case, the semiautomaton $\mathcal{C}_\eta(D) := \langle \overline{X}, \overline{U}, \overline{f} \rangle$ and the automaton $\mathcal{C}_\eta(S) := \langle \overline{X}, \overline{U}, \overline{f}, [x_0]_{\sim_X}, \overline{Y}, \overline{h} \rangle$ are well-defined and they are called the *η-canonical semiautomaton of* D and the *η-canonical automaton* of S, respectively.

Definition 83. An *encoder* is a continuous function $\mathrm{enc} : \Sigma \to U$ where Σ is a finite alphabet called *input alphabet*. An *decoder* is a continuous function $\mathrm{dec} : Y \to \Gamma$ where Γ is a finite alphabet called *output alphabet*.

Definition 84 (Definition 1 of [37])**.** Given an encoder $\mathrm{enc} : \Sigma \to U$ and a decoder $\mathrm{dec} : Y \to \Gamma$, we say that a system S *implements a function* $F : \Sigma^+ \to \Gamma$ *with encoder* enc *and decoder* dec if $F(w) = \mathrm{dec}\big(S(\mathrm{enc}(w))\big)$, for every $w \in \Sigma^+$, where $\mathrm{enc}(w) \in U^+$ applies enc element-wise. We also say that S *can-implement* F if it implements F for some choice of enc and dec. When $\Gamma = \{0,1\}$, we say that S *recognises* a language L if it implements its indicator function $\mathbb{I}_L$, and that S *can-recognise* L if it can-implement its characteristic function $\mathbb{I}_L$.

Theorem 13 (Theorem 1 of [37])**.** *An η-finite system S can-implement the same functions as its canonical automaton, which are necessarily regular.*

Definition 85 (Definition 23 of [37])**.** Given η-finite dynamics D, D', dynamics D' are a realisation of D if $\mathcal{M}(\mathcal{C}_\eta(D'))$ is a realisation of $\mathcal{M}(\mathcal{C}_\eta(D))$ of D. Given automata A, A', automaton A' is a realisation of A if the associated machine $M_{A'}$ is a realisation of of the associated machine M_A via an assignment (α, ι, ζ), and the respective initial states x'_0, x_0 are such that $x'_0 \in \alpha(x_0)$. Given an η-finite system S' and a system S, system S' is a realisation of S if $\mathcal{C}_\eta(S')$ is a realisation of $\mathcal{C}_\eta(S)$.

End of Definition 85.

Lemma 3 (Lemma 29 of [37])**.** *Suppose that semiautomaton D' is a realisation of semiautomaton D. Then (i) for any machine M with dynamics D, the canonical machine $\mathcal{M}(D')$ of D' is a realisation of M; (ii) for any automaton A with dynamics D, an initial state can be picked for $\mathcal{M}(D')$ such that the resulting automaton is a realisation of A.*

Theorem 14 (Theorem 30 of [37])**.** *Let S and S' be η-finite systems. If $\mathcal{C}_\eta(S')$ is a realisation of $\mathcal{C}_\eta(S)$, then S' can implement all the functions that S can implement.*

Theorem 15 (Theorem 31 of [37])**.** *Let D, D' be η-finite dynamics. Suppose that D' is a realisation of D. Then any function implemented by a system with dynamics D can be implemented by some system S' with dynamics D' and an identity output function.*

In the above Theorem 15, the remark on the identity output function is not present in the original statement of the theorem, but it is clear from its proof.

A.19.1 Aperiodicity and eta-convergence

Definition 86 (Convergence, Definition 31 [37])**.** For a η-finite space X, we say a sequence $(x_n)_{n\geq 1}$ with $x_n \in X$ η-converges in X, if eventually all its terms lie in the same η-component of X.

Definition 87 (Aperiodicity, Definition 32 [37])**.** We say that η-finite dynamics $D = \langle X, U, f\rangle$ are *aperiodic* if, for every $x_0 \in X$ and every input sequence $(u_n)_{n\geq 1}$ that is η-convergent in U, we have that the corresponding state sequence $(x_n)_{n\geq 1}$ is η-convergent in X.

Theorem 16 (Theorem 50 of [37])**.** *η-finite dynamics are aperiodic if and only if their η-canonical semiautomaton is group-free.*

A.20 Definite Languages

Definition 88 (Definite languages, cf. [59])**.** A language L over a finite alphabet Σ is a *definite language* if

$$L = L_1 \cup \{uv \mid u \in \Sigma^*,\, v \in L_2\}$$

for some finite languages L_1 and L_2 over Σ. We denote the class of definite languages by Def.

End of Definition 88.

Lemma 4 (Fixed-suffix characterisation of definite languages)**.** *Let L be a language over a finite alphabet Σ. Then L is definite if and only if there exists $N \in \mathbb{N}^{>0}$ such that, for every $v, w \in \Sigma^*$ and $z \in \Sigma^N$,*

$$vz \in L \iff wz \in L. \tag{2}$$

End of Lemma 4.

Proof. Suppose first that L is definite. Then there exist finite languages L_1, L_2 such that

$$L = L_1 \cup \{uv \mid u \in \Sigma^*,\, v \in L_2\}.$$

Since L_1 and L_2 are finite, there exists $N \in \mathbb{N}^{>0}$ larger than the length of every string in L_1 and at least as large as the length of every string in L_2. Let $v, w \in \Sigma^*$ and $z \in \Sigma^N$. Since $|vz|, |wz| \geq N$, neither vz nor wz belongs to L_1. Furthermore, membership in $\{ps \mid p \in \Sigma^*,\, s \in L_2\}$ is determined by a suffix s whose length is at most N, and therefore is determined completely by the common suffix z. Hence

$$vz \in L \iff wz \in L.$$

Conversely, suppose that $N \in \mathbb{N}^{>0}$ satisfies Equation (2). Define

$$L_1 := L \cap \Sigma^{*\leq N-1} \qquad L_2 := L \cap \Sigma^N.$$

Since Σ is finite, both L_1 and L_2 are finite.

Let $x \in \Sigma^*$ have length at least N and write

$$x = vz, \qquad z \in \Sigma^N.$$

Taking w to be the empty string in Equation (2) gives

$$x \in L \quad \Longleftrightarrow \quad z \in L \quad \Longleftrightarrow \quad z \in L_2.$$

Therefore

$$x \in L \quad \Longleftrightarrow \quad x \in \Sigma^* L_2.$$

The words of length strictly less than N are accounted for by L_1. Hence

$$L = L_1 \cup \{ps \mid p \in \Sigma^*,\ s \in L_2\},$$

and thus L is definite. □

A.21 Graphs, Isomorphisms, and Renaming of Nodes

We introduce the notions of graph isomorphism and equality up to a renaming of nodes that we use for computation graphs. Since the edges of a computation graph have an ordered vector of source nodes, we work with directed multi-edge graphs whose edges have the same form.

Definition 89 (Graph)**.** A *graph* is a pair

$$G = (V, E),$$

where:

- V is a finite set of *nodes*;
- $E \subseteq V^* \times V$ is a set of *directed multi-edges*.

For an edge

$$\big((u_1, \ldots, u_k), v\big) \in E,$$

we write

$$(u_1, \ldots, u_k) \to v.$$

The vector $(u_1, \ldots, u_k)$ is the *source vector* of the edge, v is its *target*, and k is its *arity*.

The ordering of the source vector is part of the graph structure: in general,

$$(u_1, u_2) \to v \qquad \text{and} \qquad (u_2, u_1) \to v$$

are distinct edges.

The *dependency relation* of G is the binary relation $\prec_G \subseteq V \times V$ defined by

$$u \prec_G v \quad \Longleftrightarrow \quad \exists (u_1, \ldots, u_k) \to v \in E \text{ such that } u = u_i \text{ for some } i \in [1..k].$$

The graph G is *acyclic* if the transitive closure of $\prec_G$ is irreflexive. *End of Definition 89.*

Definition 90 (Renaming of nodes)**.** Let

$$G = (V, E)$$

be a graph and let

$$\rho : V \to V'$$

be a bijection. We extend ρ componentwise to node-vectors by setting

$$\rho(u_1, \ldots, u_k) := \big(\rho(u_1), \ldots, \rho(u_k)\big).$$

The *renaming of G induced by ρ* is the graph

$$\rho(G) := \big(V', \rho(E)\big),$$

where

$$\rho(E) := \{\rho(u_1, \ldots, u_k) \to \rho(v) \mid (u_1, \ldots, u_k) \to v \in E\}.$$

Thus, a renaming changes only the identities of the nodes and leaves the complete edge structure, including the ordering of the source nodes of every edge, unchanged. *End of Definition 90.*

Definition 91 (Graph isomorphism)**.** Let

$$G = (V, E), \qquad G' = (V', E')$$

be graphs. A *graph isomorphism from G to G'* is a bijection

$$\rho : V \to V'$$

such that

$$(u_1, \ldots, u_k) \to v \in E \quad \iff \quad \big(\rho(u_1), \ldots, \rho(u_k)\big) \to \rho(v) \in E'$$

for every $k \in \mathbb{N}$, $(u_1, \ldots, u_k) \in V^k$, and $v \in V$.

The graphs G and G' are *isomorphic*, denoted

$$G \cong G',$$

if there exists a graph isomorphism from G to G'. *End of Definition 91.*

Proposition 5. *Let G and G' be graphs. Then,*

$$G \cong G' \quad \iff \quad \exists\, \rho \text{ a bijection between their node sets such that } \rho(G) = G'.$$

Proof. Immediate by Definitions 90 and 91. □

Remark 12. Accordingly, we say that two graphs are *equal up to a renaming of nodes* precisely when they are isomorphic. Thus,

$$G \cong G'$$

means that G and G' have exactly the same graph structure and differ at most in the names assigned to their nodes. *End of Remark 12.*

Part II

Preliminary Notions and Results

B Additional on Dynamics and Systems

Table of Contents

B.1 Parametric Dynamical Systems 59
B.2 Composition Operators for Dynamics and Systems 60
B.3 Composition Operators for Parametric Dynamics and Systems 61
B.4 Properties of Composition Operators for Parametric Dynamics and Systems 65
B.5 Euclidean Dynamics and Systems 69
B.6 State Functions of Dynamics and Systems 70

B.1 Parametric Dynamical Systems

Definition 92. *Parametric dynamics* are a tuple $D = \langle X, U, f; \Theta \rangle$ where:

- X, U, and Θ are metric spaces called the *state space*, *input space*, and *parameter space* of D, respectively;
- $f : X \times U \times \Theta \to X$ is a continuous function called the *dynamics function* of D.

For $\theta \in \Theta$, the *instance of D with parameter values θ* is the dynamics

$$D[\theta] := \langle X, U, f[\theta] \rangle, \quad \text{where} \quad f[\theta](x, u) := f(x, u; \theta).$$

End of Definition 92.

Definition 93. A *parametric (dynamical) system* is a tuple $S = \langle X, U, f, x^{\text{init}}, Y, h; \Theta_f, \Theta_h \rangle$ where:

- X, U, and Y are metric spaces called the *state space*, *input space*, and *output space* of S, respectively;
- Θ_f and Θ_h are metric spaces called the *dynamics-function parameter space* and *output-specific parameter space* of S, respectively; and $\Theta := \Theta_f \times \Theta_h$ is the parameter space of S;
- $f : X \times U \times \Theta_f \to X$ is a continuous function called the *dynamics function* of S;
- $x^{\text{init}} \in X$ is called the *initial state* of S;
- $h : X \times U \times X \times \Theta_h \to Y$ is a continuous function called the *output function* of S.

We call $\text{dyn}(S) := \langle X, U, f; \Theta_f \rangle$ the *dynamics of S*.

For $\theta_f \in \Theta_f$ and $\theta_h \in \Theta_h$, the *instance of S with parameter values (θ_f, θ_h)* is

$$S[\theta_f, \theta_h] := \langle X, U, f[\theta_f], x^{\text{init}}, Y, h[\theta_h] \rangle,$$

where

$$f[\theta_f](x, u) := f(x, u; \theta_f), \qquad h[\theta_h](x, u, x') := h\big(x, u, x'; \theta_h\big).$$

End of Definition 93.

Fact 24. *In the context of Definition 93, we have* $\text{dyn}(S[\theta_f, \theta_h]) = \text{dyn}(S)[\theta_f]$. End of Fact 24.

Definition 94 (Input sequences)**.** Let D be parametric dynamics and let S be a parametric system.

An *input sequence for D* is an input sequence for its instances $D[\theta]$, and an *input sequence for S* is an input sequence for its instances $S[\theta]$. *End of Definition 94.*

Definition 95 (State sequences)**.** Let $D = \langle X, U, f; \Theta \rangle$ be parametric dynamics, let $\theta \in \Theta$, let u be an input sequence for D, and let $x_0 \in X$.

The *state sequence of D on input u when started from x_0 with parameters θ* is the state sequence of $D[\theta]$ on input u when started from x_0.

Parametric dynamics D define the function

$$\check{D} : X \times U^* \times \Theta \to X$$

given by

$$\check{D}(x_0, u; \theta) := D\check{[}\theta](x_0, u).$$

When D are the dynamics of a parametric system S having x_0 as its initial state, *the state sequence of S with dynamics-function parameters θ* is the state sequence of D on input u when started from x_0 with parameters θ. *End of Definition 95.*

Definition 96 (Output sequences)**.** Let $S = \langle X, U, f, x^{\text{init}}, Y, h; \Theta \rangle$ be a parametric system, let $\theta \in \Theta$, and let u be an input sequence for S. The *output sequence of S on input u with parameter values θ* is the output sequence of $S[\theta]$ on input u. Parametric system S defines the function

$$\check{S} : U^+ \times \Theta \to Y, \qquad \check{S}(u; \theta) := S\check{[}\theta](u).$$

End of Definition 96.

B.2 Composition Operators for Dynamics and Systems

Definition 97 (Cascade composition of dynamics, Part I)**.** Given a pair of dynamics D_1 and D_2 along with a continuous function g,

$$D_1 = \langle X_1, U, f_1 \rangle, \qquad D_2 = \langle X_2, Z, f_2 \rangle, \qquad g : X_1 \times U \times X_1 \to Z,$$

the *cascade composition of D_1 and D_2 with connection g* is the dynamics

$$D_1 \ltimes_g D_2 := \langle X_1 \times X_2, U, f \rangle,$$

having dynamics function

$$f\big(\langle x_1, x_2 \rangle, u\big) := \big\langle x_1', f_2(x_2, z) \big\rangle \quad \text{for} \quad x_1' := f_1(x_1, u) \quad \text{and} \quad z := g(x_1, u, x_1').$$

End of Definition 97.

Definition 98 (Cascade composition of dynamics, Part II)**.** Given dynamics

$$D_1 = \langle X_1, U, f_1 \rangle, \qquad D_2 = \langle X_2, U \times X_1, f_2 \rangle,$$

their *cascade composition* are the dynamics

$$D_1 \ltimes D_2 := D_1 \ltimes_\pi D_2 \qquad \text{where} \qquad \pi(x_1, u, x_1') := \langle u, x_1 \rangle.$$

End of Definition 98.

Definition 99 (Parallel composition of dynamics)**.** Given dynamics

$$D_1 = \langle X_1, U, f_1 \rangle, \qquad D_2 = \langle X_2, U, f_2 \rangle,$$

their *parallel composition* are the dynamics

$$D_1 \parallel D_2 := \langle X_1 \times X_2, U, f \rangle,$$

having dynamics function

$$f\big(\langle x_1, x_2 \rangle, u\big) := \big\langle f_1(x_1, u), f_2(x_2, u) \big\rangle.$$

End of Definition 99.

Fact 25 (Parallel composition as a cascade)**.** *Let*

$$D_1 = \langle X_1, U, f_1 \rangle, \qquad D_2 = \langle X_2, U, f_2 \rangle$$

be dynamics.

Define

$$\widetilde{D}_2 := \langle X_2, U \times X_1, \widetilde{f}_2 \rangle,$$

where

$$\widetilde{f}_2\big(x_2, \langle u, x_1 \rangle\big) := f_2(x_2, u).$$

Then

$$D_1 \parallel D_2 = D_1 \ltimes \widetilde{D}_2.$$

End of Fact 25.

Proof. Both sides have state space $X_1 \times X_2$ and input space U. Moreover, for every $x_1 \in X_1$, $x_2 \in X_2$, and $u \in U$, their dynamics map $\langle\langle x_1, x_2\rangle, u\rangle$ to

$$\big\langle f_1(x_1, u), f_2(x_2, u)\big\rangle.$$

Hence the two dynamics are equal. □

Definition 100 (Cascade composition of systems)**.** Given systems

$$S_1 = \langle X_1, U, f_1, x_1^{\text{init}}, Z, h_1\rangle, \qquad S_2 = \langle X_2, Z, f_2, x_2^{\text{init}}, Y, h_2\rangle,$$

their *cascade composition* is the system

$$S_1 \ltimes S_2 := \big\langle X_1 \times X_2, U, f, \langle x_1^{\text{init}}, x_2^{\text{init}}\rangle, Y, h\big\rangle,$$

having dynamics function

$$f\big(\langle x_1, x_2\rangle, u\big) := \big\langle x_1', f_2(x_2, z)\big\rangle \quad \text{for} \quad x_1' := f_1(x_1, u) \quad \text{and} \quad z := h_1(x_1, u, x_1'),$$

and output function

$$h\big(\langle x_1, x_2\rangle, u, \langle x_1', x_2'\rangle\big) := h_2(x_2, z, x_2') \quad \text{for} \quad z := h_1(x_1, u, x_1').$$

End of Definition 100.

Fact 26 (Dynamics induced by layer output functions)**.** *For every pair of systems*

$$S_1 = \langle X_1, U, f_1, x_1^{\text{init}}, Z, g\rangle, \qquad S_2 = \langle X_2, Z, f_2, x_2^{\text{init}}, Y, h\rangle,$$

we have

$$\text{dyn}(S_1 \ltimes S_2) = D_1 \ltimes_g D_2.$$

End of Fact 26.

B.3 Composition Operators for Parametric Dynamics and Systems

B.3.1 Cascade Composition with Connecting Functions

Definition 101 (Cascade composition of parametric dynamics with a connection)**.** Given parametric dynamics

$$D_1 = \langle X_1, U, f_1; \Theta_1\rangle, \qquad D_2 = \langle X_2, Z, f_2; \Theta_2\rangle,$$

and a parametric connecting function

$$g : X_1 \times U \times X_1 \times \Theta_g \to Z,$$

the *cascade composition of D_1 and D_2 with connection g* is the parametric dynamics

$$D_1 \ltimes_g D_2 := \langle X_1 \times X_2, U, f; \Theta_1 \times \Theta_g \times \Theta_2\rangle,$$

having dynamics function

$$f\big(\langle x_1, x_2\rangle, u; \theta_1, \theta_g, \theta_2\big) := \big\langle x_1', x_2'\big\rangle,$$

where

$$\begin{aligned} x_1' &:= f_1(x_1, u; \theta_1), \\ z &:= g\big(x_1, u, x_1'; \theta_g\big), \\ x_2' &:= f_2(x_2, z; \theta_2). \end{aligned}$$

End of Definition 101.

B.3.2 Simple Cascade Composition

Definition 102 (Simple cascade composition of parametric dynamics)**.** Given parametric dynamics

$$D_1 = \langle X_1, U, f_1; \Theta_1 \rangle, \qquad D_2 = \langle X_2, X_1 \times U \times X_1, f_2; \Theta_2 \rangle,$$

their *simple cascade composition* is the parametric dynamics

$$D_1 \ltimes D_2 := \langle X_1 \times X_2, U, f; \Theta_1 \times \Theta_2 \rangle,$$

having dynamics function

$$f\big(\langle x_1, x_2 \rangle, u; \theta_1, \theta_2\big) := \big\langle x_1', x_2' \big\rangle,$$

where

$$\begin{aligned} x_1' &:= f_1(x_1, u; \theta_1), \\ x_2' &:= f_2(x_2, \langle x_1, u, x_1' \rangle; \theta_2). \end{aligned}$$

We also write $f = f_1 \ltimes f_2$. *End of Definition 102.*

Remark 13. A simple cascade is a special case of a cascade with a connection. In fact, let

$$\pi : X_1 \times U \times X_1 \to X_1 \times U \times X_1$$

be the identity connecting function

$$\pi(x_1, u, x_1') := \langle x_1, u, x_1' \rangle.$$

Then

$$D_1 \ltimes D_2 = D_1 \ltimes_\pi D_2.$$

End of Remark 13.

Definition 103 (Cascade composition of parametric systems)**.** Given a pair of parametric systems

$$S_1 = \langle X_1, U, f_1, x_1^{\mathrm{init}}, Z, h_1; \Theta_{f_1}, \Theta_{h_1} \rangle, \qquad S_2 = \langle X_2, Z, f_2, x_2^{\mathrm{init}}, Y, h_2; \Theta_{f_2}, \Theta_{h_2} \rangle,$$

and a choice

$$z^{\mathrm{init}} \in Z,$$

their *cascade composition* is the parametric system

$$S_1 \ltimes S_2 := \big\langle X_1 \times Z \times X_2, U, f, \langle x_1^{\mathrm{init}}, z^{\mathrm{init}}, x_2^{\mathrm{init}} \rangle, Y, h; \Theta_f, \Theta_h \big\rangle,$$

having parameter spaces

$$\Theta_f := \Theta_{f_1} \times \Theta_{h_1} \times \Theta_{f_2}, \qquad \Theta_h := \Theta_{h_2},$$

dynamics function

$$f\big(\langle x_1, \bar{z}, x_2 \rangle, u; \theta_{f_1}, \theta_{h_1}, \theta_{f_2}\big) := \big\langle x_1', z, x_2' \big\rangle,$$

and output function

$$h\big(\langle x_1, \bar{z}, x_2 \rangle, u, \langle x_1', z, x_2' \rangle; \theta_{h_2}\big) := h_2\big(x_2, z, x_2'; \theta_{h_2}\big),$$

where

$$\begin{aligned} x_1' &:= f_1(x_1, u; \theta_{f_1}), \\ z &:= h_1\big(x_1, u, x_1'; \theta_{h_1}\big), \\ x_2' &:= f_2(x_2, z; \theta_{f_2}). \end{aligned}$$

End of Definition 103.

Remark 14 (Stored connecting value). The additional state component in

$$X_1 \times Z \times X_2$$

stores the connecting value produced during the current transition. Its previous value $\bar{z}$ is not used by the dynamics and is overwritten at every step.

Consequently, the choice of z^{init} does not affect any state component other than the unused connecting component at time zero, and does not affect the input–output behaviour on nonempty input words.

Storing the connecting value makes the cascade compatible with Definition 93: the composite output function depends only on its output-specific parameters $\Theta_h = \Theta_{h_2}$, while the parameters Θ_{h_1} required to generate the connecting value belong only to the composite dynamics-function parameter space Θ_f. *End of Remark 14.*

Fact 27 (Dynamics induced by parametric layer output functions)**.** *For every pair of parametric systems S_1 and S_2 as in Definition 103, the output function*

$$h_1 : X_1 \times U \times X_1 \times \Theta_{h_1} \to Z$$

is a parametric connecting function with parameter space Θ_{h_1}.

In particular, let

$$\widehat{D} := \mathrm{dyn}(S_1) \ltimes_{h_1} \mathrm{dyn}(S_2) = \langle X_1 \times X_2, U, \widehat{f}; \Theta_{f_1} \times \Theta_{h_1} \times \Theta_{f_2} \rangle.$$

Then, for every

$$x_1 \in X_1, \qquad \bar{z} \in Z, \qquad x_2 \in X_2, \qquad u \in U,$$

and every admissible parameter value, if

$$\widehat{f}\big(\langle x_1, x_2 \rangle, u; \theta_{f_1}, \theta_{h_1}, \theta_{f_2}\big) = \langle x_1', x_2' \rangle,$$

then

$$f\big(\langle x_1, \bar{z}, x_2 \rangle, u; \theta_{f_1}, \theta_{h_1}, \theta_{f_2}\big) = \big\langle x_1', h_1\big(x_1, u, x_1'; \theta_{h_1}\big), x_2' \big\rangle.$$

Thus the first and third state components evolve exactly according to the connected cascade $\mathrm{dyn}(S_1) \ltimes_{h_1} \mathrm{dyn}(S_2)$, while the middle component stores its connecting value. End of Fact 27.

Proposition 6 (Dynamics composition commutes with instantiation)**.** *Let*

$$D_1 = \langle X_1, U, f_1; \Theta_1 \rangle, \qquad D_2 = \langle X_2, Z, f_2; \Theta_2 \rangle,$$

and let

$$g : X_1 \times U \times X_1 \times \Theta_g \to Z$$

be a parametric connecting function.

Then:

$$(D_1 \ltimes_g D_2)[\theta_1, \theta_g, \theta_2] = D_1[\theta_1] \ltimes_{g[\theta_g]} D_2[\theta_2],$$
$$(D_1 \parallel D_2)[\theta_1, \theta_2] = D_1[\theta_1] \parallel D_2[\theta_2].$$

Proof. Immediate from the definitions. □

B.3.3 Parallel Composition

Definition 104 (Parallel composition of parametric dynamics)**.** Given a pair of parametric dynamics

$$D_1 = \langle X_1, U, f_1; \Theta_1 \rangle, \qquad D_2 = \langle X_2, U, f_2; \Theta_2 \rangle,$$

their *parallel composition* is the parametric dynamics

$$D_1 \parallel D_2 := \langle X_1 \times X_2, U, f; \Theta_1 \times \Theta_2 \rangle,$$

having dynamics function

$$f\big(\langle x_1, x_2 \rangle, u; \theta_1, \theta_2\big) := \big\langle f_1(x_1, u; \theta_1), f_2(x_2, u; \theta_2) \big\rangle.$$

We also write $f = f_1 \parallel f_2$. *End of Definition 104.*

Fact 28 (Parallel composition as a cascade)**.** *Let*

$$D_1 = \langle X_1, U, f_1; \Theta_1 \rangle, \qquad D_2 = \langle X_2, U, f_2; \Theta_2 \rangle$$

be parametric dynamics.

Define

$$\widetilde{D}_2 := \langle X_2, X_1 \times U \times X_1, \widetilde{f}_2; \Theta_2 \rangle,$$

where

$$\widetilde{f}_2(x_2, \langle x_1, u, x_1' \rangle; \theta_2) := f_2(x_2, u; \theta_2).$$

Then

$$D_1 \parallel D_2 = D_1 \ltimes \widetilde{D}_2.$$

End of Fact 28.

Proof. Both sides have state space $X_1 \times X_2$, input space U, and parameter space $\Theta_1 \times \Theta_2$. Moreover, for every $x_1 \in X_1$, $x_2 \in X_2$, $u \in U$, $\theta_1 \in \Theta_1$, and $\theta_2 \in \Theta_2$, letting

$$x_1' := f_1(x_1, u; \theta_1),$$

the simple cascade has dynamics

$$\big\langle x_1', \widetilde{f}_2(x_2, \langle x_1, u, x_1' \rangle; \theta_2) \big\rangle = \big\langle f_1(x_1, u; \theta_1), f_2(x_2, u; \theta_2) \big\rangle,$$

which is precisely the dynamics of $D_1 \parallel D_2$. □

B.3.4 Input Precomposition

Definition 105 (Parametric input precomposition)**.** Let

$$D = \langle X, Z, f; \Theta \rangle$$

be parametric dynamics, and let

$$q : V \times \Theta_q \to Z$$

be a parametric continuous function.

The *input precomposition of D by q* is the parametric dynamics

$$q \triangleright D := \langle X, V, f_q; \Theta_q \times \Theta \rangle,$$

having dynamics function

$$f_q\big(x, v; \theta_q, \theta\big) := f(x, q(v; \theta_q); \theta).$$

End of Definition 105.

B.3.5 Output Extensions

Definition 106 (Output extension of parametric dynamics)**.** Let

$$D \;=\; \big\langle X,\, U,\, f;\; \Theta_f \big\rangle$$

be parametric dynamics, let $x_0 \in X$, let Y and Θ_h be metric spaces, and let

$$h \;:\; (X \times U \times X) \times \Theta_h \;\to\; Y$$

be a continuous function.

The *output extension of $\langle D, x_0 \rangle$ by h* is the parametric system

$$\langle D, x_0 \rangle \triangleright h \;:=\; \big\langle X,\, U,\, f,\, x_0,\, Y,\, h;\; \Theta_f,\, \Theta_h \big\rangle.$$

We call it an *analytic output extension* if h is an analytic Euclidean function. *End of Definition 106.*

Definition 107 (Output extension of systems)**.** Given a system and a continuous function, given respectively by

$$S \;=\; \big\langle X,\, U,\, f,\, x_0,\, Y,\, h \big\rangle\,, \qquad\qquad \tilde{h} : Y \times \to \tilde{Y}\,,$$

the *output extension of S by $\tilde{h}$* is the system

$$S \triangleright \tilde{h} \;:=\; \big\langle X,\, U,\, f,\, x_0,\, Y,\, \tilde{h} \circ h \big\rangle\,.$$

We call it an *analytic output extension* if $\tilde{h}$ is an analytic Euclidean function. *End of Definition 107.*

Definition 108 (Output extension of parametric systems)**.** Given a parametric system and a continuous parametric function, given respectively by

$$S \;=\; \big\langle X,\, U,\, f,\, x_0,\, Y,\, h;\; \Theta_f,\, \Theta_h \big\rangle\,, \qquad\qquad \tilde{h} : Y \times \Theta_{\tilde{h}} \to \tilde{Y}\,,$$

the *output extension of S by $\tilde{h}$* is the parametric system

$$S \triangleright \tilde{h} \;:=\; \big\langle X,\, U,\, f,\, x_0,\, Y,\; \tilde{h} \circ h\,;\; \Theta_f,\, \Theta_h \times \Theta_{\tilde{h}} \big\rangle\,.$$

We call it an *analytic output extension* if $\tilde{h}$ is an analytic Euclidean function. *End of Definition 108.*

Fact 29 (Parametric systems as output extensions)**.** *Every parameteric system*

$$S \;=\; \big\langle X,\, U,\, f,\, x_0,\, Y,\, h; \Theta_f,\, \Theta_h \big\rangle\,, \qquad\qquad \tilde{h} : Y \times \Theta_{\tilde{h}} \to \tilde{Y}\,.$$

can be expressed as the output extension

$$S \;=\; \langle S, x_0 \rangle \triangleright h\,.$$

End of Fact 29.

B.4 Properties of Composition Operators for Parametric Dynamics and Systems

Proposition 7 (Dynamics composition commutes with instantiation)**.** *Let*

$$D_1 = \langle X_1, U, f_1; \Theta_1 \rangle, \qquad D_2 = \langle X_2, Z, f_2; \Theta_2 \rangle,$$

and let

$$g : X_1 \times U \times X_1 \times \Theta_g \to Z$$

be a parametric connecting function.

Then:

(1) for every $\theta_1 \in \Theta_1$, $\theta_g \in \Theta_g$, and $\theta_2 \in \Theta_2$,

$$(D_1 \ltimes_g D_2)[\theta_1, \theta_g, \theta_2] = D_1[\theta_1] \ltimes_{g[\theta_g]} D_2[\theta_2]; \tag{3}$$

(2) if

$$Z = X_1 \times U \times X_1,$$

then, for every $\theta_1 \in \Theta_1$ and $\theta_2 \in \Theta_2$,

$$(D_1 \ltimes D_2)[\theta_1, \theta_2] = D_1[\theta_1] \ltimes D_2[\theta_2]; \tag{4}$$

(3) if $Z = U$, then, for every $\theta_1 \in \Theta_1$ and $\theta_2 \in \Theta_2$,

$$(D_1 \parallel D_2)[\theta_1, \theta_2] = D_1[\theta_1] \parallel D_2[\theta_2]. \tag{5}$$

Proof. For Equation (3), fix

$$\theta_1 \in \Theta_1, \qquad \theta_g \in \Theta_g, \qquad \theta_2 \in \Theta_2.$$

By definition, the dynamics function of

$$(D_1 \ltimes_g D_2)[\theta_1, \theta_g, \theta_2]$$

maps $\langle \langle x_1, x_2 \rangle, u \rangle$ to

$$\big\langle x_1', f_2(x_2, z; \theta_2) \big\rangle,$$

where

$$x_1' := f_1(x_1, u; \theta_1), \qquad z := g\big(x_1, u, x_1'; \theta_g\big).$$

Since

$$g[\theta_g](x_1, u, x_1') = g\big(x_1, u, x_1'; \theta_g\big),$$

this is exactly the dynamics function of

$$D_1[\theta_1] \ltimes_{g[\theta_g]} D_2[\theta_2].$$

For Equation (4), both sides map $\langle \langle x_1, x_2 \rangle, u \rangle$ to

$$\big\langle x_1', f_2(x_2, \langle x_1, u, x_1' \rangle; \theta_2) \big\rangle, \qquad x_1' := f_1(x_1, u; \theta_1).$$

Finally, Equation (5) follows because both sides map $\langle \langle x_1, x_2 \rangle, u \rangle$ to

$$\big\langle f_1(x_1, u; \theta_1), f_2(x_2, u; \theta_2) \big\rangle.$$

□

Proposition 8 (System cascade composition commutes with instantiation)**.** *Let*

$$S_1 = \langle X_1, U, f_1, x_1^{\text{init}}, Z, h_1; \Theta_{f_1}, \Theta_{h_1} \rangle, \qquad S_2 = \langle X_2, Z, f_2, x_2^{\text{init}}, Y, h_2; \Theta_{f_2}, \Theta_{h_2} \rangle,$$

and fix the same $z^{\text{init}} \in Z$ in the parametric and nonparametric cascade constructions.

Then, for every

$$\theta_{f_1} \in \Theta_{f_1}, \qquad \theta_{h_1} \in \Theta_{h_1}, \qquad \theta_{f_2} \in \Theta_{f_2}, \qquad \theta_{h_2} \in \Theta_{h_2},$$

we have

$$(S_1 \ltimes S_2)[\langle \theta_{f_1}, \theta_{h_1}, \theta_{f_2} \rangle, \theta_{h_2}] = S_1[\theta_{f_1}, \theta_{h_1}] \ltimes S_2[\theta_{f_2}, \theta_{h_2}]. \tag{6}$$

Proof. Fix admissible parameter values. Both sides have state space

$$X_1 \times Z \times X_2,$$

input space U, initial state

$$\langle x_1^{\mathrm{init}}, z^{\mathrm{init}}, x_2^{\mathrm{init}} \rangle,$$

and output space Y.

Let

$$\langle x_1, \bar{z}, x_2 \rangle \in X_1 \times Z \times X_2, \qquad u \in U.$$

On both sides,

$$\begin{aligned} x_1' &:= f_1(x_1, u; \theta_{f_1}), \\ z &:= h_1\big(x_1, u, x_1'; \theta_{h_1}\big), \\ x_2' &:= f_2(x_2, z; \theta_{f_2}), \end{aligned}$$

and hence the dynamics function maps

$$\langle \langle x_1, \bar{z}, x_2 \rangle, u \rangle$$

to

$$\langle x_1', z, x_2' \rangle.$$

Likewise, on both sides the output function maps

$$\big\langle \langle x_1, \bar{z}, x_2 \rangle, u, \langle x_1', z, x_2' \rangle \big\rangle$$

to

$$h_2\big(x_2, z, x_2'; \theta_{h_2}\big).$$

Therefore the two instantiated systems coincide. □

Corollary 4 (Underlying dynamics of a system cascade)**.** *Let S_1 and S_2 be parametric systems as above, and fix*

$$\theta_{f_1} \in \Theta_{f_1}, \qquad \theta_{h_1} \in \Theta_{h_1}, \qquad \theta_{f_2} \in \Theta_{f_2}, \qquad \theta_{h_2} \in \Theta_{h_2}.$$

Let

$$\widehat{D} := \mathrm{dyn}\big(S_1[\theta_{f_1}, \theta_{h_1}]\big) \ltimes_{h_1[\theta_{h_1}]} \mathrm{dyn}\big(S_2[\theta_{f_2}, \theta_{h_2}]\big),$$

and let $\widehat{f}$ denote its dynamics function.

Let $\bar{f}$ denote the dynamics function of

$$\mathrm{dyn}\big((S_1 \ltimes S_2)[\langle \theta_{f_1}, \theta_{h_1}, \theta_{f_2} \rangle, \theta_{h_2}]\big).$$

Then, for every

$$x_1 \in X_1, \qquad \bar{z} \in Z, \qquad x_2 \in X_2, \qquad u \in U,$$

if

$$\widehat{f}\big(\langle x_1, x_2 \rangle, u\big) = \langle x_1', x_2' \rangle,$$

then

$$\bar{f}\big(\langle x_1, \bar{z}, x_2 \rangle, u\big) = \big\langle x_1', h_1[\theta_{h_1}]\big(x_1, u, x_1'\big), x_2' \big\rangle. \tag{7}$$

In particular, the first and third state components evolve exactly as $\widehat{D}$.

Proof. By Proposition 8, the instantiated system cascade uses

$$\begin{aligned} x_1' &= f_1(x_1, u; \theta_{f_1}), \\ z &= h_1[\theta_{h_1}]\big(x_1, u, x_1'\big), \\ x_2' &= f_2(x_2, z; \theta_{f_2}). \end{aligned}$$

The pair $\langle x_1', x_2' \rangle$ is precisely the update produced by $\widehat{D}$, while the additional middle component stores z. This gives Equation (7). □

Corollary 5 (Simple dynamics cascades induced by system cascades)**.** *Let S_1 and S_2 be as above, and suppose that*

$$h_1\big(x_1, u, x_1'; \theta_{h_1}\big) = \langle x_1, u, x_1' \rangle \tag{8}$$

for every admissible

$$x_1, \qquad u, \qquad x_1', \qquad \theta_{h_1}.$$

Then the first and third state components of

$$\mathrm{dyn}\big((S_1 \ltimes S_2)[\langle \theta_{f_1}, \theta_{h_1}, \theta_{f_2} \rangle, \theta_{h_2}]\big)$$

evolve exactly according to the simple cascade

$$\mathrm{dyn}\big(S_1[\theta_{f_1}, \theta_{h_1}]\big) \ltimes \mathrm{dyn}\big(S_2[\theta_{f_2}, \theta_{h_2}]\big).$$

Proof. Under Equation (8), the instantiated connecting function $h_1[\theta_{h_1}]$ is the identity connecting function

$$\pi(x_1, u, x_1') := \langle x_1, u, x_1' \rangle.$$

Hence the connected cascade appearing in Corollary 4 is precisely the simple cascade. □

Corollary 6 (Instantiation of iterated compositions)**.** *Let*

$$D_1, \ldots, D_n$$

be composable parametric dynamics with parameter spaces

$$\Theta_1, \ldots, \Theta_n,$$

and let

$$\theta = \langle \theta_1, \ldots, \theta_n \rangle \in \Theta_1 \times \cdots \times \Theta_n.$$

Then

$$(D_1 \ltimes \cdots \ltimes D_n)[\theta] = D_1[\theta_1] \ltimes \cdots \ltimes D_n[\theta_n].$$

Likewise, whenever parallel composition is well-defined,

$$(D_1 \parallel \cdots \parallel D_n)[\theta] = D_1[\theta_1] \parallel \cdots \parallel D_n[\theta_n].$$

The analogous identity holds for iterated cascade compositions of parametric systems, using the same initial connecting values in the parametric and instantiated cascades and the canonical identification of finite Cartesian products.

Proof. The claims follow by induction on n from associativity, Proposition 7, and Proposition 8. □

Remark 15 (Generality of simple cascades). Proposition 9 and Theorem 17 show that, whenever the considered class of parametric recurrent dynamics is closed under parametric input precomposition, allowing arbitrary parametric connecting functions does not enlarge the class of cascade dynamics: every connected cascade can be represented exactly as a simple cascade. *End of Remark 15.*

B.4.1 Generality of Simple Cascades

Proposition 9 (Absorption of connecting functions)**.** *Let*

$$D_1 = \langle X_1, U, f_1; \Theta_1 \rangle, \qquad D_2 = \langle X_2, Z, f_2; \Theta_2 \rangle,$$

and let

$$g : X_1 \times U \times X_1 \times \Theta_g \to Z$$

be a parametric connecting function.

Let

$$\widetilde{D}_2 := g \triangleright D_2,$$

where g is regarded as a parametric map from

$$X_1 \times U \times X_1$$

to Z. Explicitly,

$$\widetilde{D}_2 = \langle X_2, X_1 \times U \times X_1, \widetilde{f}_2; \Theta_g \times \Theta_2 \rangle,$$

where

$$\widetilde{f}_2\big(x_2, \langle x_1, u, x_1' \rangle; \theta_g, \theta_2\big) := f_2(x_2, g\big(x_1, u, x_1'; \theta_g\big); \theta_2). \tag{9}$$

Then

$$D_1 \ltimes_g D_2 = D_1 \ltimes \widetilde{D}_2. \tag{10}$$

Proof. Both sides have state space

$$X_1 \times X_2,$$

input space U, and parameter space

$$\Theta_1 \times \Theta_g \times \Theta_2.$$

For every

$$x_1 \in X_1, \qquad x_2 \in X_2, \qquad u \in U,$$

and every

$$\theta_1 \in \Theta_1, \qquad \theta_g \in \Theta_g, \qquad \theta_2 \in \Theta_2,$$

let

$$x_1' := f_1(x_1, u; \theta_1).$$

The second component of the connected cascade is

$$f_2(x_2, g\big(x_1, u, x_1'; \theta_g\big); \theta_2).$$

By Equation (9), this is exactly

$$\widetilde{f}_2\big(x_2, \langle x_1, u, x_1' \rangle; \theta_g, \theta_2\big),$$

which is the second component of the simple cascade. The first components are identical by definition. □

Definition 109 (Closure under parametric input precomposition)**.** A class $\mathcal{D}$ of parametric dynamics is *closed under input precomposition* with parametric functions from a class $\mathcal{F}^{\mathrm{p}}$ if, for every

$$D \in \mathcal{D}$$

and every parametric continuous function $f \in \mathcal{F}^{\mathrm{p}}$ for which $f \triangleright D$ is well-defined, we have

$$\big(f \triangleright D\big) \in \mathcal{D}.$$

End of Definition 109.

Theorem 17 (Generality of simple cascades)**.** *Let $\mathcal{D}$ be a class of parametric dynamics closed under input precomposition with parametric functions from a class $\mathcal{F}^{\mathrm{p}}$.*

Let D be a cascade given by

$$D \;=\; D_1 \ltimes_g D_2$$

with $D_2 \in \mathcal{D}$ and connecting function $g \in \mathcal{F}^{\mathrm{p}}$. Then D is equal to a simple cascade

$$\tilde{D} \;=\; D_1 \ltimes \tilde{D}_2$$

for some $\tilde{D}_2 \in \mathcal{D}$.

Consequently, the class of simple cascades having units from $\mathcal{D}$ coincides with the class of cascades having units from $\mathcal{D}$ and connecting functions from $\mathcal{F}^{\mathrm{p}}$.

Proof. By Proposition 9,

$$D_1 \ltimes_g D_2 = D_1 \ltimes \big(g \triangleright D_2\big).$$

Since $\mathcal{D}$ is closed under parametric input precomposition and $D_2 \in \mathcal{D}$, we have

$$\big(g \triangleright D_2\big) \in \mathcal{D}.$$

Taking

$$\tilde{D}_2 := g \triangleright D_2$$

proves the claim. □

Fact 30 (Composition of recurrent-unit input functions)**.** *We note that cascade compositions employing connecting functions can be expressed as simple cascade compositions—which do not employ connecting functions.*

Consider a class $\mathcal{D}^{\mathrm{p}}$ of parametric recurrent units

$$D \;=\; \big\langle X, U, f; \Theta_f \times \Theta_\varphi \,|\, \varphi \big\rangle \qquad \textit{with} \qquad f(x, u; \theta_f, \theta_\varphi) \;=\; \tilde{f}\big(x, \varphi(u; \theta_\varphi); \theta_f\big),\ \varphi \in \mathcal{F}^{\mathrm{p}}.$$

Namely, the dependence of the dynamics function f on inputs u is specified through a parametric function $\varphi \in \mathcal{F}^{\mathrm{p}}$. For such class $\mathcal{D}^{\mathrm{p}}$, closure under input precomposition with parametric functions from Ψ amounts to closure of $\mathcal{F}^{\mathrm{p}}$ under precomposition with functions from Ψ. In fact, for every $D \in \mathcal{D}^{\mathrm{p}}$ and every $\psi \in \Psi$, we have

$$D_\psi \;:=\; \psi \triangleright D \;=\; \big\langle X, U, f; \Theta_f \times \Theta_\varphi \times \Theta_\psi \,|\, \tilde{\varphi} \big\rangle \quad \textit{with} \quad \tilde{\varphi}(u; \theta_\varphi, \theta_\psi) \;=\; \varphi\big(\psi(u; \theta_\psi); \theta_\varphi\big),$$

and hence $D_\psi \in \mathcal{D}^{\mathrm{p}}$.

Thus, all dynamics D given by a cascade composition of units from $\mathcal{D}^{\mathrm{p}}$ with connecting functions from $\mathcal{F}^{\mathrm{p}}$, can be expressed as a simple cascade composition of units from $\mathcal{D}^{\mathrm{p}}$—with no use of connecting functions. End of Fact 30.

Fact 31 (Associativity)**.** *Under the canonical identification of finite Cartesian products, the cascade and parallel composition operators for parametric dynamics are associative. The cascade composition operator for parametric systems is likewise associative when the initial connecting values are fixed for each inter-layer space.*

In particular, the state space of a finite system cascade is canonically identified with the interleaved product

$$X_1 \times Z_1 \times X_2 \times \cdots \times Z_{n-1} \times X_n,$$

independently of parenthesisation. Hence one can write

$$D_1 \ltimes \cdots \ltimes D_n, \qquad D_1 \parallel \cdots \parallel D_n, \qquad S_1 \ltimes \cdots \ltimes S_n$$

without specifying parentheses. End of Fact 31.

B.5 Euclidean Dynamics and Systems

Definition 110 (Euclidean parametric dynamics and systems)**.** Let us consider parametric dynamics $D = \langle X, U, f; \Theta \rangle$. Then, we say that dynamics D are *Euclidean* if the ambient spaces Ω_X, Ω_U, and Ω_Θ of X, U, and Θ are Euclidean spaces—namely, $X \subseteq \Omega_X = \mathbb{R}^n$, $U \subseteq \Omega_U = \mathbb{R}^m$, and $\Theta \subseteq \Omega_\Theta = \mathbb{R}^p$ for some $n, m, p \in \mathbb{N}^{>0}$; *End of Definition 110.*

Definition 111 (Euclidean and measurable-Euclidean parametric systems)**.** Let us consider a parametric system $S = \langle X, U, f, x^{\mathrm{init}}, Y, h; \Theta_f, \Theta_h \rangle$. Then, we say that S is *Euclidean* if its dynamics $\mathrm{dyn}(S)$ are Euclidean and additionally the ambient spaces Ω_Y and Ω_{Θ_h} of Y and Θ_h are Euclidean spaces—namely, $Y \subseteq \Omega_Y = \mathbb{R}^q$ and $\Theta_h \subseteq \Omega_{\Theta_h} = \mathbb{R}^r$ for some $q, r \in \mathbb{N}^{>0}$; *End of Definition 111.*

B.5.1 Analytic Output Extensions

Definition 112 (Analytic output extension)**.** Let

$$D = \langle X, U, f; \Theta_f \rangle$$

be Euclidean parametric dynamics, let $x^{\mathrm{init}} \in X$, let Y and Θ_h be Euclidean spaces, and let

$$h \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}(X \times U \times X, Y; \Theta_h).$$

The output extension

$$(D, x^{\mathrm{init}}) \triangleright h$$

is called the *analytic output extension of* (D, x^{init}) *by* h. *End of Definition 112.*

B.6 State Functions of Dynamics and Systems

Definition 113. Given dynamics $D = \langle X, U, f \rangle$, their *state-to-state function* for fixed $u \in U^\omega$ and $t \in \mathbb{T}$ is the function $\check{D}^u_t : X \to X$ that maps each state $x_0 \in X$ to the corresponding state at time t, namely

$$\check{D}^u_t(x_0) \;:=\; \check{D}(x_0, u_{[1..t]})\,.$$

End of Definition 113.

Definition 114. Given parametric dynamics $D = \langle X, U, f; \Theta \rangle$, their *state-to-state function* for fixed $u \in U^\omega$, $\theta \in \Theta$, and $t \in \mathbb{T}$ is the function $\check{D}[u, t; \theta] : X \to X$ that maps each state $x_0 \in X$ to the corresponding state at time t,

$$\check{D}[u, t; \theta](x_0) \;:=\; \check{D}(x_0, u_{[1..t]}; \theta)\,.$$

End of Definition 114.

Definition 115. Given parametric dynamics $D = \langle X, U, f; \Theta \rangle$,

- their *parameter-to-state function* at time $t \in \mathbb{T}$ for a fixed *infinite* input sequence $u \in U^\omega$ and a fixed initial state $x_0 \in X$, is the function $\check{D}[x_0, u, t] : \Theta \to X$ that maps each parameter value $\theta \in \Theta$ to the state of D at time t when executed on input u starting in state x_0,

$$\check{D}[x_0, u, t](\theta) \;:=\; \check{D}(x_0, u_{[1..t]}; \theta)\,.$$

- their *parameter-to-latest-state function*, for a fixed *finite* input sequence $u \in U^+$ of length t and a fixed initial state $x_0 \in X$, is the function $\check{D}[x_0, u] : \Theta \to X$ that maps each parameter value $\theta \in \Theta$ to the state of D at time t when executed on input u starting in state x_0,

$$\check{D}[x_0, u](\theta) \;:=\; \check{D}(x_0, u; \theta)\,;$$

Given a parametric system S having dynamics $D := \mathrm{dyn}(S)$ and initial state x_0, its *parameter-to-latest-state function*, for a fixed *finite* input sequence $u \in U^+$ is the function

$$\check{S}[u] \;:=\; \check{D}[x_0, u]\,.$$

End of Definition 115.

Definition 116 (Intermediate state-to-state function)**.** Let us consider dynamics $D = \langle X, U, f \rangle$, an input sequence $u \in U^\omega$, and indices $s, t \in \mathbb{N}$ with $s < t$. Then, the *state-to-state function of D from time s to time t* on input u, is the function

$$\check{[}u][s\!:\!t] : X \to X$$

that maps each state $x \in X$ at time s to the state reached at time t under the input subsequence $u_{[s+1..t]}$, namely

$$\check{[}u][s\!:\!t][x] \;:=\; \check{D}\big(x, u_{[s+1..t]}\big)\,.$$

End of Definition 116.

C Generic Architectures

Table of Contents

C.1 Recurrent Layers 71
C.2 Recurrent Cascades 72
C.3 Recurrent Neural Cascades (RNCs) 73
C.4 Remarks on Convolution 76

C.1 Recurrent Layers

Definition 117 (Recurrent layer presentations)**.** We define recurrent layer presentations and their parametric counterparts.

▸ **SubDefinition 117.1**. A *recurrent layer presentation* is an expression

$$\mathsf{L} \;:=\; D_1 \parallel \cdots \parallel D_n$$

where $n \in \mathbb{N}^{>0}$ is called the *width* of the presentation, and $D_1, \ldots, D_n$ are dynamics having a common input space, called the *units* of the presentation.

The dynamics *presented* by L are

$$[\![\mathsf{L}]\!] \;:=\; D_1 \parallel \cdots \parallel D_n\,.$$

We also say that L is a presentation of the recurrent layer $[\![\mathsf{L}]\!]$.

For $\mathcal{D}$ a class of dynamics, the presentation L is a *recurrent layer of type* $\mathcal{D}$ if its units belong to $\mathcal{D}$, namely $\{D_1, \ldots, D_n\} \subseteq \mathcal{D}$. *End of SD.117.1* ◂

▸ **SubDefinition 117.2**. A *parametric recurrent layer presentation* is an expression

$$\mathsf{L}^{\mathrm{p}} \;:=\; D_1^{\mathrm{p}} \parallel \cdots \parallel D_n^{\mathrm{p}}$$

where $n \in \mathbb{N}^{>0}$ is called the *width* of the presentation, and $D_1^{\mathrm{p}}, \ldots, D_n^{\mathrm{p}}$ are parametric dynamics having a common input space, called the *units* of the presentation.

The parametric dynamics *presented* by L^{p} are

$$[\![\mathsf{L}^{\mathrm{p}}]\!] \;:=\; D_1^{\mathrm{p}} \parallel \cdots \parallel D_n^{\mathrm{p}}\,.$$

We also say that L^{p} is a presentation of the parametric recurrent layer $[\![\mathsf{L}^{\mathrm{p}}]\!]$.

For $\mathcal{D}^{\mathrm{p}}$ a class of parametric dynamics, the presentation L^{p} is a *parameteric recurrent layer presentation of type* $\mathcal{D}^{\mathrm{p}}$ if its units belong to $\mathcal{D}^{\mathrm{p}}$—namely $\{D_1^{\mathrm{p}}, \ldots, D_n^{\mathrm{p}}\} \subseteq \mathcal{D}^{\mathrm{p}}$. *End of SD.117.2* ◂

End of Definition 117.

Definition 118 (Recurrent layer dynamics)**.** We define recurrent layer dynamics along with their parametric counterparts.

▸ **SubDefinition 118.1**. *Recurrent layer dynamics* are dynamics that are presented by some recurrent layer presentation. *End of SD.118.1* ◂

▸ **SubDefinition 118.2**. *Parametric recurrent layer dynamics* are parametric dynamics that are presented by some parametric recurrent layer presentation. *End of SD.118.2* ◂

End of Definition 118.

Definition 119 (Classes of recurrent layer dynamics)**.** We define classes of recurrent layer dynamics along with their parametric counterparts.

▸ **SubDefinition 119.1**. Let $\mathcal{D}$ be a class of dynamics.

We say that recurrent layer dynamics are *recurrent layer $\mathcal{D}$-dynamics* if they are presented by some recurrent layer presentation of type $\mathcal{D}$. *End of SD.119.1* ◂

▸ **SubDefinition 119.2**. Let $\mathcal{D}^{\mathrm{p}}$ be a class of parametric dynamics.

We say that parametric recurrent layer dynamics are *parametric recurrent layer $\mathcal{D}^{\mathrm{p}}$-dynamics* if they are presented by some parametric recurrent layer presentation of type $\mathcal{D}^{\mathrm{p}}$. *End of SD.119.2* ◂

End of Definition 119.

C.2 Recurrent Cascades

Definition 120 (Recurrent cascade presentations)**.** We define recurrent cascade presentations and their parametric counterparts.

▸ **SubDefinition 120.1**. A *recurrent cascade presentation* is an expression

$$\mathsf{C} \;:=\; g_1 \blacktriangleright \mathsf{L}_1 \blacktriangleright\!\!\!\prec_{g_2} \mathsf{L}_2 \blacktriangleright\!\!\!\prec_{g_3} \cdots \blacktriangleright\!\!\!\prec_{g_\ell} \mathsf{L}_\ell$$

where $\ell \in \mathbb{N}^{>0}$ is called the *depth* of the presentation, $\mathsf{L}_1, \ldots, \mathsf{L}_\ell$ are recurrent layer presentations, called the *layers* of the presentation, and $g_1, \ldots, g_\ell$ are continuous functions, called the *connecting functions* of the presentation, such that the composition

$$[\![\mathsf{C}]\!] \;:=\; g_1 \triangleright [\![\mathsf{L}_1]\!] \ltimes_{g_2} [\![\mathsf{L}_2]\!] \ltimes_{g_3} \cdots \ltimes_{g_\ell} [\![\mathsf{L}_\ell]\!]$$

is well-defined; in this case, we call $[\![\mathsf{C}]\!]$ the dynamics *presented* by C.

We also say that C is a presentation of the recurrent cascade dynamics $[\![\mathsf{C}]\!]$.

Letting n_i be the width of the layer presentation L_i, for each $i \in [1..\ell]$, we call

$$(n_1, \ldots, n_\ell)$$

the *width profile* of the presentation C.

For $\mathcal{D}$ a class of dynamics and $\mathcal{F}$ a class of functions, the presentation C is a *recurrent cascade presentation of type* $(\mathcal{D}, \mathcal{F})$ if (i) each of its layers is a recurrent layer presentation of type $\mathcal{D}$, and (ii) each of its connecting functions belongs to $\mathcal{F}$—namely $\{g_1, \ldots, g_\ell\} \subseteq \mathcal{F}$. *End of SD.120.1* ◂

▸ **SubDefinition 120.2**. A *parametric recurrent cascade presentation* is an expression

$$\mathsf{C}^{\mathrm{p}} \;:=\; g_1^{\mathrm{p}} \blacktriangleright \mathsf{L}_1^{\mathrm{p}} \blacktriangleright\!\!\!\prec_{g_2^{\mathrm{p}}} \mathsf{L}_2^{\mathrm{p}} \blacktriangleright\!\!\!\prec_{g_3^{\mathrm{p}}} \cdots \blacktriangleright\!\!\!\prec_{g_\ell^{\mathrm{p}}} \mathsf{L}_\ell^{\mathrm{p}}$$

where $\ell \in \mathbb{N}^{>0}$ is called the *depth* of the presentation, $\mathsf{L}_1^{\mathrm{p}}, \ldots, \mathsf{L}_\ell^{\mathrm{p}}$ are parametric recurrent layer presentations, called the *layers* of the presentation, and $g_1^{\mathrm{p}}, \ldots, g_\ell^{\mathrm{p}}$ are parametric continuous functions, called the *connecting functions* of the presentation, such that the composition

$$[\![\mathsf{C}^{\mathrm{p}}]\!] \;:=\; g_1^{\mathrm{p}} \triangleright [\![\mathsf{L}_1^{\mathrm{p}}]\!] \ltimes_{g_2^{\mathrm{p}}} [\![\mathsf{L}_2^{\mathrm{p}}]\!] \ltimes_{g_3^{\mathrm{p}}} \cdots \ltimes_{g_\ell^{\mathrm{p}}} [\![\mathsf{L}_\ell^{\mathrm{p}}]\!]$$

is well-defined; in this case, we call $[\![\mathsf{C}^{\mathrm{p}}]\!]$ the parametric dynamics *presented* by C^{p}.

We also say that C^{p} is a presentation of the parametric recurrent cascade dynamics $[\![\mathsf{C}^{\mathrm{p}}]\!]$.

Letting n_i be the width of $\mathsf{L}_i^{\mathrm{p}}$, for each $i \in [1..\ell]$, we call

$$(n_1, \ldots, n_\ell)$$

the *width profile* of the presentation C.

For $\mathcal{D}^{\mathrm{p}}$ a class of parametric dynamics and $\mathcal{F}^{\mathrm{p}}$ a class of parametric functions, the presentation C^{p} is a *parametric recurrent cascade presentation of type* $(\mathcal{D}^{\mathrm{p}}, \mathcal{F}^{\mathrm{p}})$ if (i) each of its layers is a parametric recurrent layer presentation of type $\mathcal{D}^{\mathrm{p}}$, and (ii) each of its connecting functions belongs to $\mathcal{F}^{\mathrm{p}}$—namely $\{g_1^{\mathrm{p}}, \ldots, g_\ell^{\mathrm{p}}\} \subseteq \mathcal{F}^{\mathrm{p}}$. *End of SD.120.2* ◂

End of Definition 120.

Definition 121 (Recurrent cascade dynamics)**.** We define recurrent cascade dynamics along with their parametric counterparts.

- **SubDefinition 121.1**. Dynamics are called *recurrent cascade dynamics* if they are presented by some recurrent cascade presentation. *End of SD.121.1* ◂

- **SubDefinition 121.2**. Parametric dynamics are called *parametric recurrent cascade dynamics* if they are presented by some parametric recurrent cascade presentation. *End of SD.121.2* ◂

End of Definition 121.

Definition 122 (Classes of recurrent cascade dynamics)**.** We define classes of recurrent cascade dynamics along with their parametric counterparts.

- **SubDefinition 122.1**. Given a class $\mathcal{D}$ of dynamics and a class $\mathcal{F}$ of functions, recurrent cascade dynamics are called *recurrent cascade dynamics with units from $\mathcal{D}$ and connecting functions from $\mathcal{F}$* if they are presented by some recurrent cascade presentation of type $(\mathcal{D}, \mathcal{F})$. *End of SD.122.1* ◂

- **SubDefinition 122.2**. Given a class $\mathcal{D}^{\mathrm{p}}$ of parametric dynamics and a class $\mathcal{F}^{\mathrm{p}}$ of parametric functions, parametric recurrent cascade dynamics are called *parametric recurrent cascade dynamics with units from $\mathcal{D}^{\mathrm{p}}$ and connecting functions from $\mathcal{F}^{\mathrm{p}}$* if they are presented by some recurrent cascade presentation of type $(\mathcal{D}^{\mathrm{p}}, \mathcal{F}^{\mathrm{p}})$. *End of SD.122.2* ◂

End of Definition 122.

C.3 Recurrent Neural Cascades (RNCs)

Definition 123 (Recurrent Neural Cascade Dynamics)**.** We define recurrent neural cascade dynamics and their parametric counterpart.

- **SubDefinition 123.1**. Let $\mathcal{D}$ be a class of Euclidean dynamics.

  The *class of Recurrent Neural Cascade dynamics of type $\mathcal{D}$* consists of all recurrent cascade dynamics whose units are from $\mathcal{D}$ and whose connecting functions are from $\mathcal{F}_{\mathrm{AN}}$.

  It is abbreviated as *$\mathcal{D}$-RNC dynamics*.

  It is denoted by $\mathrm{RNC}_{\mathrm{dyn}}(\mathcal{D})$. *End of SD.123.1* ◂

- **SubDefinition 123.2**. Let $\mathcal{D}^{\mathrm{p}}$ be a class of parametric Euclidean dynamics.

  The *class of parametric Recurrent Neural Cascade dynamics of type $\mathcal{D}^{\mathrm{p}}$* consists of all parametric recurrent cascade dynamics whose units are from $\mathcal{D}^{\mathrm{p}}$ and whose connecting functions are from $\mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}$.

  It is abbreviated as *$\mathcal{D}^{\mathrm{p}}$-RNC dynamics*.

  It is denoted by $\mathrm{RNC}^{\mathrm{p}}_{\mathrm{dyn}}(\mathcal{D}^{\mathrm{p}})$. *End of SD.123.2* ◂

End of Definition 123.

Definition 124 (Recurrent Neural Cascades)**.** We define recurrent neural cascades and their parametric counterpart.

- **SubDefinition 124.1**. Let $\mathcal{D}$ be a class of Euclidean dynamics.

  The *class of Recurrent Neural Cascades (RNCs) of type $\mathcal{D}$* consists of each system S having dynamics $\mathrm{dyn}(S) \in \mathrm{RNC}_{\mathrm{dyn}}(\mathcal{D})$ and output function $h \in \mathcal{F}_{\mathrm{AN}}$.

  It is abbreviated as *$\mathcal{D}$-RNCs*.

  It is denoted by $\mathrm{RNC}(\mathcal{D})$. *End of SD.124.1* ◂

- **SubDefinition 124.2**. Let $\mathcal{D}^{\mathrm{p}}$ be a class of parametric Euclidean dynamics.

The *class of parametric Recurrent Neural Cascades (RNCs) of type* $\mathcal{D}^{\mathrm{p}}$ consists of each parametric system S^{p} having parametric dynamics $\mathrm{dyn}(S^{\mathrm{p}}) \in \mathrm{RNC}^{\mathrm{p}}_{\mathrm{dyn}}(\mathcal{D}^{\mathrm{p}})$ and parametric output function $h^{\mathrm{p}} \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}$.

It is abbreviated as $\mathcal{D}^{\mathrm{p}}$-*RNCs*.

It is denoted by $\mathrm{RNC}^{\mathrm{p}}(\mathcal{D}^{\mathrm{p}})$. *End of SD.124.2* ◂

End of Definition 124.

Definition 125 (RNC-closed dynamics class)**.** We define the notion of RNC-closed class.

A class $\mathcal{D}$ of dynamics is *RNC-closed* if $\mathrm{RNC}_{\mathrm{dyn}}(\mathcal{D}) \subseteq \mathcal{D}$.

A class $\mathcal{D}^{\mathrm{p}}$ of parametric dynamics is *RNC-closed* if $\mathrm{RNC}^{\mathrm{p}}_{\mathrm{dyn}}(\mathcal{D}^{\mathrm{p}}) \subseteq \mathcal{D}^{\mathrm{p}}$.

C.3.1 RNCs as analytic output extension presentations

Definition 126 (Output extension presentations)**.** We define output extension presentations and their parametric counterparts.

▸ **SubDefinition 126.1**. Let D be a recurrent layer or cascade presentation, let $D := [\![\mathsf{D}]\!]$ be the dynamics presented by D, let X be the state space of D, let U be the input space of D, let $x^{\mathrm{init}} \in X$, and let $h : X \times U \times X \to Y$ be a continuous function.

The expression

$$\mathsf{S} \;:=\; \langle \mathsf{D}, x^{\mathrm{init}} \rangle \blacktriangleright h$$

is called the *output extension presentation of* $(\mathsf{D}, x^{\mathrm{init}})$ *by* h, and the system *presented* by S is

$$[\![\mathsf{S}]\!] \;:=\; \langle [\![\mathsf{D}]\!], x^{\mathrm{init}} \rangle \triangleright h\,.$$

We also call S a presentation of the system $[\![\mathsf{S}]\!]$.

We say that S is an *analytic output extension presentation* if dynamics D are Euclidean and $h \in \mathcal{F}_{\mathrm{AN}}$. *End of SD.126.1* ◂

▸ **SubDefinition 126.2**. Let D^{p} be a parametric recurrent layer or cascade presentation, let $D^{\mathrm{p}} := [\![\mathsf{D}^{\mathrm{p}}]\!]$ be the parametric dynamics presented by D^{p}, let X be the state space of D^{p}, let U be the input space of D^{p}, let $x^{\mathrm{init}} \in X$, and let $h^{\mathrm{p}} : (X \times U \times X) \times \Theta_h \to Y$ be a parametric continuous function—where the parameter space is Θ_h.

The expression

$$\mathsf{S}^{\mathrm{p}} \;:=\; \langle \mathsf{D}^{\mathrm{p}}, x^{\mathrm{init}} \rangle \blacktriangleright h^{\mathrm{p}}$$

is called the *parametric output extension presentation of* $(\mathsf{D}^{\mathrm{p}}, x^{\mathrm{init}})$ *by* h^{p}, and the parametric system *presented* by S^{p} is

$$[\![\mathsf{S}^{\mathrm{p}}]\!] \;:=\; \langle [\![\mathsf{D}^{\mathrm{p}}]\!], x^{\mathrm{init}} \rangle \triangleright h^{\mathrm{p}}\,.$$

We also call S^{p} a presentation of the parametric system $[\![\mathsf{S}^{\mathrm{p}}]\!]$.

We say that S^{p} is a *parametric analytic output extension presentation* if parametric dynamics D^{p} are Euclidean and $h^{\mathrm{p}} \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}$. *End of SD.126.2* ◂

End of Definition 126.

Fact 32 (RNCs as analytic output extension presentations)**.** *Let $\mathcal{D}$ be a class of Euclidean dynamics, and let $S \in \mathrm{RNC}(\mathcal{D})$ with*

$$S \;=\; \big\langle X, U, f, x^{\mathrm{init}}, Y, h \big\rangle\,.$$

There exists a recurrent cascade presentation C of the dynamics $\mathrm{dyn}(S)$ such that S is presented by the output extension presentation

$$\mathsf{S} \;:=\; \langle \mathsf{C}, x^{\mathrm{init}} \rangle \blacktriangleright h\,;$$

namely,

$$\mathrm{dyn}(S) \;=\; [\![\mathsf{C}]\!] \qquad \textit{and} \qquad S \;=\; [\![\mathsf{S}]\!] \;=\; \langle [\![\mathsf{C}]\!], x^{\mathrm{init}} \rangle \triangleright h\,;$$

in particular, the recurrent cascade presentation C *has type* $(\mathcal{D}, \mathcal{F}_{\mathrm{AN}})$*, and the output extension* S *is an analytic output extension.* End of Fact 32.

Fact 33 (Parametric RNCs as analytic output extension presentations)**.** *Let* $\mathcal{D}^{\mathrm{p}}$ *be a class of parametric Euclidean dynamics, and let* $S^{\mathrm{p}} \in \mathrm{RNC}^{\mathrm{p}}(\mathcal{D}^{\mathrm{p}})$ *with*

$$S^{\mathrm{p}} \;=\; \left\langle X, U, f^{\mathrm{p}}, x^{\mathrm{init}}, Y, h^{\mathrm{p}}; \Theta_f, \Theta_h \right\rangle .$$

There exists a parametric recurrent cascade presentation C^{p} *of the parametric dynamics* $\mathrm{dyn}(S^{\mathrm{p}})$ *such that* S^{p} *is presented by the parametric output extension presentation*

$$\mathsf{S}^{\mathrm{p}} \;:=\; \langle \mathsf{C}^{\mathrm{p}}, x^{\mathrm{init}} \rangle \blacktriangleright h^{\mathrm{p}}\,;$$

namely,

$$\mathrm{dyn}(S^{\mathrm{p}}) \;=\; [\![\mathsf{C}^{\mathrm{p}}]\!] \qquad \textit{and} \qquad S \;=\; [\![\mathsf{S}^{\mathrm{p}}]\!] \;=\; \langle [\![\mathsf{C}^{\mathrm{p}}]\!], x^{\mathrm{init}} \rangle \triangleright h^{\mathrm{p}}\,;$$

in particular, the parametric recurrent cascade presentation C^{p} *has type* $(\mathcal{D}^{\mathrm{p}}, \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}})$*, and the output extension* S^{p} *is an analytic output extension.* End of Fact 33.

Proof of Facts 32, 33. We first prove Fact 32.

Let $S \in \mathrm{RNC}(\mathcal{D})$. By Definition 124, we have

$$\mathrm{dyn}(S) \in \mathrm{RNC}_{\mathrm{dyn}}(\mathcal{D}) \qquad \text{and} \qquad h \in \mathcal{F}_{\mathrm{AN}}.$$

By Definition 123 and Definition 122, there exists a recurrent cascade presentation C of type $(\mathcal{D}, \mathcal{F}_{\mathrm{AN}})$ such that

$$[\![\mathsf{C}]\!] = \mathrm{dyn}(S).$$

Hence, by Definition 126, the expression

$$\mathsf{S} \;:=\; \langle \mathsf{C}, x^{\mathrm{init}} \rangle \blacktriangleright h$$

is an output extension presentation satisfying

$$[\![\mathsf{S}]\!] \;=\; \langle [\![\mathsf{C}]\!], x^{\mathrm{init}} \rangle \triangleright h \;=\; \langle \mathrm{dyn}(S), x^{\mathrm{init}} \rangle \triangleright h \;=\; S\,.$$

Since $\mathrm{dyn}(S)$ is Euclidean and $h \in \mathcal{F}_{\mathrm{AN}}$, Definition 126 further implies that S is an analytic output extension presentation. This proves Fact 32.

The parametric claim is analogous.

Let $S^{\mathrm{p}} \in \mathrm{RNC}^{\mathrm{p}}(\mathcal{D}^{\mathrm{p}})$.

By Definition 124, we have

$$\mathrm{dyn}(S^{\mathrm{p}}) \in \mathrm{RNC}^{\mathrm{p}}_{\mathrm{dyn}}(\mathcal{D}^{\mathrm{p}}) \qquad \text{and} \qquad h^{\mathrm{p}} \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}\,.$$

By Definition 123 and Definition 122, there exists a parametric recurrent cascade presentation C^{p} of type $(\mathcal{D}^{\mathrm{p}}, \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}})$ such that

$$[\![\mathsf{C}^{\mathrm{p}}]\!] = \mathrm{dyn}(S^{\mathrm{p}})\,.$$

Therefore, by Definition 126, the expression

$$\mathsf{S}^{\mathrm{p}} \;:=\; \langle \mathsf{C}^{\mathrm{p}}, x^{\mathrm{init}} \rangle \blacktriangleright h^{\mathrm{p}}$$

is a parametric output extension presentation satisfying

$$[\![\mathsf{S}^{\mathrm{p}}]\!] \;=\; \langle [\![\mathsf{C}^{\mathrm{p}}]\!], x^{\mathrm{init}} \rangle \triangleright h^{\mathrm{p}} \;=\; \langle \mathrm{dyn}(S^{\mathrm{p}}), x^{\mathrm{init}} \rangle \triangleright h^{\mathrm{p}} \;=\; S^{\mathrm{p}}\,.$$

Since $\mathrm{dyn}(S^{\mathrm{p}})$ is Euclidean and $h^{\mathrm{p}} \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}$, Definition 126 further implies that S^{p} is a parametric analytic output extension presentation. This proves Fact 33. □

C.4 Remarks on Convolution

We show that convolutional preprocessing can be expressed naturally through cascade composition. More precisely, a convolution acting on a two-step window can be realised as the output function of a system having a trivial state, and prepending this convolution to arbitrary target dynamics is equivalent to precomposing the target dynamics by the convolution map.

Definition 127 (Two-step convolution)**.** Let $d_{\mathrm{cv}} \in \mathbb{N}^{>0}$, and let

$$U \subseteq \Omega_U = \mathbb{R}^{d_{\mathrm{in}}}, \qquad \Theta_{\mathrm{cv}} := \mathbb{R}^{d_{\mathrm{cv}} \times d_{\mathrm{in}}} \times \mathbb{R}^{d_{\mathrm{cv}} \times d_{\mathrm{in}}} \times \mathbb{R}^{d_{\mathrm{cv}}}.$$

The *two-step convolution* with input space U and parameter space Θ_{cv} is the parametric Euclidean map $\mathrm{conv}_{U,\Theta_{\mathrm{cv}}} : U^2 \times \Theta_{\mathrm{cv}} \to \mathbb{R}^{d_{\mathrm{cv}}}$ given by

$$\mathrm{conv}_{U,\Theta_{\mathrm{cv}}}\big(u^{\mathrm{prev}}, u^{\mathrm{cur}}; K_1, K_2, b\big) := K_1 \cdot u^{\mathrm{cur}} + K_2 \cdot u^{\mathrm{prev}} + b\,.$$

We write conv in place of $\mathrm{conv}_{U,\Theta_{\mathrm{cv}}}$ when U, Θ_{cv} are clear from the context. *End of Definition 127.*

Fact 34. *The two-step convolution of Definition 127 is a parametric analytic Euclidean map.*
End of Fact 34.

Proof. The function $\mathrm{conv}_{\Omega_U,\Omega_{\Theta_{\mathrm{cv}}}}$ is an ambient extension of $\mathrm{conv}_{U,\Theta_{\mathrm{cv}}}$, and it is polynomial in all of its arguments, and hence ordinarily-analytic. □

Definition 128 (Stateless convolution system)**.** Let $X_0 = \{x_0\}$ be a singleton space.

The *stateless system associated with the two-step convolution* of Definition 127 is the parametric system

$$S_{\mathrm{cv}} := \langle X_0, U^2, f_0, x_0, \mathbb{R}^{d_{\mathrm{cv}}}, h_{\mathrm{cv}};\ X_0, \Theta_{\mathrm{cv}}\rangle$$

where

$$f_0\big(x_0, (u^{\mathrm{prev}}, u^{\mathrm{cur}}); x_0\big) := x_0\,,$$
$$h_{\mathrm{cv}}\big(x_0, (u^{\mathrm{prev}}, u^{\mathrm{cur}}), x_0; K_1, K_2, b\big) := \mathrm{conv}\big(u^{\mathrm{prev}}, u^{\mathrm{cur}}; K_1, K_2, b\big)\,.$$

End of Definition 128.

Proposition 10 (Prepending convolution by cascade composition)**.** *Let $D = \langle X, U, f; \Theta_f\rangle$ be arbitrary parametric dynamics with $U \subseteq \mathbb{R}^{d_{\mathrm{cv}}}$, and let S_{cv} be the stateless convolution system of Definition 128. Let us define the parametric dynamics*

$$\hat{D} := \mathrm{dyn}(S_{\mathrm{cv}}) \ltimes_{h_{\mathrm{cv}}} D \qquad \textit{and} \qquad D_{\mathrm{cv}} := \mathrm{conv} \triangleright D\,.$$

We have that dynamics $\hat{D}$ and D_{cv} are isomorphic.

In particular, the above dynamics are of the form

$$\hat{D} = \langle X_0 \times X, U^2, \hat{f}; X_0, \Theta_{\mathrm{cv}}, \Theta_f\rangle, \qquad D_{\mathrm{cv}} = \langle X, U^2, f_{\mathrm{cv}}; \Theta_{\mathrm{cv}}, \Theta_f\rangle,$$

and dynamics $\hat{D}$ are isomorphic to dynamics D_{cv}, under the state-space isomorphism $\pi_2(x_0, x) := x$ and parameter-space isomorphism $\pi_{2,3}(x_0, \theta_{\mathrm{cv}}, \theta_f) := (\theta_{\mathrm{cv}}, \theta_f)$.

Namely, for every $x \in X$, every $u^{\mathrm{p,c}} \in U^2$, every $\theta_{\mathrm{cv}} \in \Theta_{\mathrm{cv}}$, and every $\theta_f \in \Theta_f$,

$$\begin{aligned}\pi_2\Big(\hat{f}\big(\langle x_0, x\rangle, u^{\mathrm{p,c}}; x_0, \theta_{\mathrm{cv}}, \theta_f\big)\Big) &= f_{\mathrm{cv}}\big(\pi_2(x_0, x), u^{\mathrm{p,c}}; \pi_{2,3}(x_0, \theta_{\mathrm{cv}}, \theta_f)\big)\\ &= f_{\mathrm{cv}}\big(x, u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}, \theta_f\big)\,.\end{aligned}$$

End of Proposition 10.

Proof. By Definition 128,

$$\mathrm{dyn}(S_{\mathrm{cv}}) = \langle X_0, U^2, f_0; X_0\rangle,$$

where

$$f_0\big(x_0, u^{\mathrm{p,c}}; x_0\big) = x_0$$

for every $u^{\mathrm{p,c}} \in U^2$, and

$$h_{\mathrm{cv}}\big(x_0, u^{\mathrm{p,c}}, x_0; \theta_{\mathrm{cv}}\big) = \mathrm{conv}(u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}).$$

Hence, by Definition 101,

$$\hat{D} = \langle X_0 \times X, U^2, \hat{f}; X_0, \Theta_{\mathrm{cv}}, \Theta_f \rangle,$$

and, for every $x \in X$, every $u^{\mathrm{p,c}} \in U^2$, every $\theta_{\mathrm{cv}} \in \Theta_{\mathrm{cv}}$, and every $\theta_f \in \Theta_f$, we have

$$\begin{aligned}\hat{f}\big(\langle x_0, x\rangle, u^{\mathrm{p,c}}; x_0, \theta_{\mathrm{cv}}, \theta_f\big) &= \big\langle f_0\big(x_0, u^{\mathrm{p,c}}; x_0\big), f\big(x, h_{\mathrm{cv}}\big(x_0, u^{\mathrm{p,c}}, f_0\big(x_0, u^{\mathrm{p,c}}; x_0\big); \theta_{\mathrm{cv}}\big); \theta_f\big)\big\rangle \\ &= \big\langle x_0, f\big(x, \mathrm{conv}(u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}); \theta_f\big)\big\rangle.\end{aligned}$$

Therefore, applying π_2 gives

$$\pi_2\left(\hat{f}\big(\langle x_0, x\rangle, u^{\mathrm{p,c}}; x_0, \theta_{\mathrm{cv}}, \theta_f\big)\right) = f\big(x, \mathrm{conv}(u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}); \theta_f\big). \tag{11}$$

On the other hand, by Definition 105,

$$D_{\mathrm{cv}} = \mathrm{conv} \triangleright D = \langle X, U^2, f_{\mathrm{cv}}; \Theta_{\mathrm{cv}}, \Theta_f \rangle,$$

with

$$f_{\mathrm{cv}}\big(x, u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}, \theta_f\big) = f\big(x, \mathrm{conv}(u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}); \theta_f\big). \tag{12}$$

Combining Equations (11) and (12), and using

$$\pi_2(x_0, x) = x, \qquad \pi_{2,3}(x_0, \theta_{\mathrm{cv}}, \theta_f) = (\theta_{\mathrm{cv}}, \theta_f),$$

we obtain

$$\begin{aligned}\pi_2\left(\hat{f}\big(\langle x_0, x\rangle, u^{\mathrm{p,c}}; x_0, \theta_{\mathrm{cv}}, \theta_f\big)\right) &= f_{\mathrm{cv}}\big(\pi_2(x_0, x), u^{\mathrm{p,c}}; \pi_{2,3}(x_0, \theta_{\mathrm{cv}}, \theta_f)\big) \\ &= f_{\mathrm{cv}}\big(x, u^{\mathrm{p,c}}; \theta_{\mathrm{cv}}, \theta_f\big).\end{aligned}$$

Finally, since X_0 is a singleton, the coordinate projections

$$\pi_2 : X_0 \times X \to X$$

and

$$\pi_{2,3} : X_0 \times \Theta_{\mathrm{cv}} \times \Theta_f \to \Theta_{\mathrm{cv}} \times \Theta_f$$

are isomorphisms. Since $\hat{D}$ and D_{cv} moreover have the same input space U^2, the preceding identity shows that their dynamics functions are intertwined by these isomorphisms. Therefore, $\hat{D}$ and D_{cv} are isomorphic. □

Corollary 7 (Prepending convolution to a parametric system)**.** *Let*

$$S = \langle X, \mathbb{R}^{d_{\mathrm{cv}}}, f, x^{\mathrm{init}}, Y, h; \Theta_f, \Theta_h \rangle$$

be a parametric system.

In the cascade

$$S_{\mathrm{cv}} \ltimes S,$$

the target-state component evolves according to the convolution-precomposed dynamics

$$\mathrm{conv} \triangleright \mathrm{dyn}(S).$$

In particular, for an input sequence

$$v = \big(u_t^{\mathrm{p,c}}\big)_{t \in [T]} = \big((u_{t-1}, u_t)\big)_{t \in [T]} \in \big(U^2\big)^+,$$

the input presented to the target system at time t is

$$\mathrm{conv}(u_t^{\mathrm{p,c}}; \theta_{\mathrm{cv}}) = K_1 \cdot u_t + K_2 \cdot u_{t-1} + b.$$

End of Corollary 7.

Proof. By Proposition 10, the connected dynamics

$$\mathrm{dyn}(S_{\mathrm{cv}}) \ltimes_{h_{\mathrm{cv}}} \mathrm{dyn}(S)$$

are isomorphic to

$$\mathrm{conv} \triangleright \mathrm{dyn}(S),$$

with the isomorphism given by projection onto the target-state component.

Hence, in the system cascade $S_{\mathrm{cv}} \ltimes S$, the target-state component evolves according to $\mathrm{conv} \triangleright \mathrm{dyn}(S)$.

For

$$u_t^{\mathrm{p,c}} = (u_{t-1}, u_t),$$

Definition 127 gives

$$\mathrm{conv}(u_t^{\mathrm{p,c}}; \theta_{\mathrm{cv}}) = K_1 \cdot u_t + K_2 \cdot u_{t-1} + b,$$

which proves the claim. □

Definition 129 (Repeat dynamics, Definition 24 of [37])**.** The *repeat dynamics on* U are the dynamics

$$R_U \;:=\; \langle U^2, U, r_U \rangle, \qquad \text{where} \qquad r_U\left((u^{\mathrm{prev}}, u^{\mathrm{cur}}), u\right) \;:=\; (u^{\mathrm{cur}}, u)\,.$$

We denote by

$$R_U^{\mathrm{p}} \;:=\; \langle U^2, U, r_U^{\mathrm{p}}; X_0 \rangle$$

its trivially-parametric counterpart, where

$$r_U^{\mathrm{p}}\big((u^{\mathrm{prev}}, u^{\mathrm{cur}}), u; x_0\big) \;:=\; (u^{\mathrm{cur}}, u)\,.$$

End of Definition 129.

Fact 35 (State sequence of repeat dynamics)**.** *Let*

$$u^{\mathrm{init}} = (u_{-1}, u_0) \in U^2$$

be an initial state of R_U*, and let*

$$u = (u_t)_{t \in [T]} \in U^+$$

be an input sequence.

Then the state sequence $(s_t)_{t \in [0..T]}$ *of* R_U *satisfies*

$$s_0 = (u_{-1}, u_0), \qquad s_t = (u_{t-1}, u_t) \qquad \forall t \in [T].$$

End of Fact 35.

Proof. The claim for $t = 0$ holds by the choice of initial state.

For $t \in [T]$, assuming

$$s_{t-1} = (u_{t-2}, u_{t-1}),$$

Definition 129 gives

$$\begin{aligned} s_t &= r_U(s_{t-1}, u_t) \\ &= r_U\left((u_{t-2}, u_{t-1}), u_t\right) \\ &= (u_{t-1}, u_t). \end{aligned}$$

The claim follows by induction. □

Definition 130 (Repeat-convolution system)**.** Fix

$$u_{-1}, u_0 \in U.$$

The *repeat-convolution system* is the parametric system

$$S_{\mathrm{rep,cv}} \;:=\; \langle U^2, U, r_U^{\mathrm{p}}, (u_{-1}, u_0), \mathbb{R}^{d_{\mathrm{cv}}}, h_{\mathrm{rep,cv}}; X_0, \Theta_{\mathrm{cv}} \rangle,$$

where

$$h_{\mathrm{rep,cv}}\big(s, u, s'; \theta_{\mathrm{cv}}\big) \;:=\; \mathrm{conv}(s'; \theta_{\mathrm{cv}}).$$

End of Definition 130.

Corollary 8 (Prepending causal convolution to an unwindowed input sequence)**.** *Let*

$$S = \langle X, \mathbb{R}^{d_{\mathrm{cv}}}, f, x^{\mathrm{init}}, Y, h; \Theta_f, \Theta_h \rangle$$

be a parametric system, and consider the cascade

$$S_{\mathrm{rep,cv}} \ltimes S.$$

For every input sequence

$$u = (u_t)_{t\in[T]} \in U^+,$$

the input presented to the target system at time t is

$$\mathrm{conv}(u_{t-1}, u_t; \theta_{\mathrm{cv}}) = K_1 \cdot u_t + K_2 \cdot u_{t-1} + b \qquad \forall t \in [T].$$

Thus, prepending the repeat-convolution system $S_{\mathrm{rep,cv}}$ realises causal two-step convolution directly on the input sequence $(u_t)_{t\in[T]}$. End of Corollary 8.

Proof. Let $(s_t)_{t\in[0..T]}$ be the state sequence of the repeat-dynamics component of $S_{\mathrm{rep,cv}}$.

By Fact 35,

$$s_t = (u_{t-1}, u_t) \qquad \forall t \in [T].$$

By Definition 130, the connecting value produced by $S_{\mathrm{rep,cv}}$ at time t is therefore

$$\begin{aligned} h_{\mathrm{rep,cv}}\big(s_{t-1}, u_t, s_t; \theta_{\mathrm{cv}}\big) &= \mathrm{conv}(s_t; \theta_{\mathrm{cv}}) \\ &= \mathrm{conv}(u_{t-1}, u_t; \theta_{\mathrm{cv}}) \\ &= K_1 \cdot u_t + K_2 \cdot u_{t-1} + b. \end{aligned}$$

By the definition of cascade composition of parametric systems, this connecting value is precisely the input presented to the target system S at time t. □

Remark 16 (Windowed and unwindowed realisations of convolution). Corollary 7 and Corollary 8 give two equivalent ways of placing a causal two-step convolution before a target system.

In Corollary 7, the external input has already been transformed into the window sequence

$$\big((u_{t-1}, u_t)\big)_{t\in[T]},$$

and consequently the convolutional system can have a trivial state.

In Corollary 8, the external input is the original sequence $(u_t)_{t\in[T]}$. The repeat dynamics R_U provide precisely the one-step memory needed to construct the state

$$(u_{t-1}, u_t)$$

at time t, after which the same convolution map conv is applied. *End of Remark 16.*

D MinMax Algebra for MinMax Recurrences

Table of Contents

D.1 Scalar MinMax Algebra 80
D.2 Componentwise MinMax Operations 81
D.3 MinMax Matrix Algebra 82
D.4 MinMax Affine Operators 83
D.5 MinMax Recurrences 84
D.6 Finite Restrictions of the MinMax Algebra 85

This appendix develops the algebraic operations underlying MinMax recurrences. They arise by specialising the lattice-theoretic and algebraic notions of Sections A.13 and A.14 to the real line equipped with its usual order.

At the scalar level, the lattice join and meet become maximum and minimum, respectively. We denote these operations by $\oplus$ and $\odot$, and extend both operations componentwise to vectors and matrices.

The scalar operations $\oplus$ and $\odot$ then induce a MinMax matrix product, denoted by $\otimes$. It is obtained from ordinary matrix multiplication by replacing scalar addition with $\oplus$ and scalar multiplication with $\odot$. Thus, $\otimes$ is reserved for the induced matrix–vector and matrix–matrix products, whereas $\odot$ denotes scalar or componentwise minimum.

D.1 Scalar MinMax Algebra

We first specialise the order-theoretic notion of lattice to the real line.

Proposition 11 (Real distributive lattice)**.** *The ordered set $(\mathbb{R}, \leq)$ is a distributive lattice, with*

$$x \wedge y = \min\{x, y\}, \qquad x \vee y = \max\{x, y\} \qquad \forall x, y \in \mathbb{R}.$$

Proof. For $x, y \in \mathbb{R}$, $\min\{x, y\}$ is a lower bound of x and y, and every common lower bound is at most $\min\{x, y\}$. Hence it is their meet. Dually, $\max\{x, y\}$ is their join.

To show distributivity, let $x, y, z \in \mathbb{R}$. Since the usual order is total, assume without loss of generality that $y \leq z$. Then

$$x \wedge (y \vee z) = x \wedge z = (x \wedge y) \vee (x \wedge z),$$

because $x \wedge y \leq x \wedge z$. The dual distributive identity follows analogously. □

We now introduce the notation used throughout the MinMax constructions.

Definition 131 (Scalar MinMax algebra)**.** The *scalar MinMax algebra* is the binary algebra

$$\mathbb{M} := (\mathbb{R}, \oplus, \odot),$$

where, for every $x, y \in \mathbb{R}$,

$$x \oplus y := x \vee y = \max\{x, y\},$$
$$x \odot y := x \wedge y = \min\{x, y\}.$$

End of Definition 131.

Thus, $\oplus$ and $\odot$ are respectively the join and meet operations of the real lattice, written using the algebraic notation employed by MinMax recurrences.

Proposition 12 (Algebraic structure of the scalar MinMax algebra)**.** *The scalar MinMax algebra $\mathbb{M}$ is a distributive lattice algebra.*

Proof. By Proposition 11, $(\mathbb{R}, \leq)$ is a distributive lattice whose join and meet are respectively maximum and minimum. Hence, by Fact 16 and Definition 131,

$$(\mathbb{R}, \oplus, \odot)$$

is the corresponding distributive lattice algebra. □

Fact 36 (Scalar MinMax identities)**.** *The operations $\oplus$ and $\odot$ are associative, commutative, idempotent, and mutually distributive.*

In particular, for every $x, y, z \in \mathbb{R}$,

$$\begin{aligned} &(x \oplus y) \oplus z = x \oplus (y \oplus z), && x \oplus y = y \oplus x, && x \oplus x = x, \\ &(x \odot y) \odot z = x \odot (y \odot z), && x \odot y = y \odot x, && x \odot x = x, \end{aligned}$$

and

$$\begin{aligned} x \odot (y \oplus z) &= (x \odot y) \oplus (x \odot z), \\ x \oplus (y \odot z) &= (x \oplus y) \odot (x \oplus z). \end{aligned}$$

Furthermore, the absorption identities give

$$\begin{aligned} x \oplus (x \odot y) &= x, \\ x \odot (x \oplus y) &= x. \end{aligned}$$

End of Fact 36.

Definition 132 (MinMax reductions)**.** Let $n \in \mathbb{N}^{>0}$ with $n \geq 2$.

The *Min reduction* of arity n is

$$\mathsf{min}_n : \mathbb{R}^n \to \mathbb{R}, \qquad \mathsf{min}_n(x_1, \ldots, x_n) := \bigodot_{i=1}^{n} x_i = \min_{i \in [1..n]} x_i,$$

The *Max reduction* of arity n is

$$\mathsf{max}_n : \mathbb{R}^n \to \mathbb{R}, \qquad \mathsf{max}_n(x_1, \ldots, x_n) := \bigoplus_{i=1}^{n} x_i = \max_{i \in [1..n]} x_i.$$

We call both min_n or max_n a *MinMax reduction*.

The *MinMax primitive function family* is

$$\mathcal{P}_{\circledast} := \big\{\mathsf{min}_n \mid n \in \mathbb{N} \text{ with } n \geq 2\big\} \cup \big\{\mathsf{max}_n \mid n \in \mathbb{N} \text{ with } n \geq 2\big\}.$$

End of Definition 132.

Proposition 13 (Finite distributivity of the scalar MinMax operations)**.** *For every $n \in \mathbb{N}^{>0}$, every $x \in \mathbb{R}$, and every $y_1, \ldots, y_n \in \mathbb{R}$,*

$$x \odot \bigoplus_{i=1}^{n} y_i = \bigoplus_{i=1}^{n} \big(x \odot y_i\big), \tag{13}$$

$$x \oplus \bigodot_{i=1}^{n} y_i = \bigodot_{i=1}^{n} \big(x \oplus y_i\big). \tag{14}$$

Proof. By Proposition 12, $\oplus$ and $\odot$ are mutually distributive. The finite identities follow by induction on n, using associativity of the two operations. □

D.2 Componentwise MinMax Operations

For every $n \in \mathbb{N}^{>0}$, we equip $\mathbb{R}^n$ with the componentwise order $\leq_{\mathrm{cw}}$ defined by

$$x \leq_{\mathrm{cw}} y \quad \Longleftrightarrow \quad (x)_i \leq (y)_i \qquad \forall i \in [1..n].$$

Proposition 14 (Euclidean distributive lattice)**.** *For every $n \in \mathbb{N}^{>0}$, the ordered set*

$$(\mathbb{R}^n, \leq_{\mathrm{cw}})$$

is a distributive lattice whose meet and join are computed componentwise:

$$\begin{aligned} (x \wedge y)_i &= (x)_i \wedge (y)_i = \min\{(x)_i, (y)_i\}, \\ (x \vee y)_i &= (x)_i \vee (y)_i = \max\{(x)_i, (y)_i\} \end{aligned}$$

for every $i \in [1..n]$.

Proof. The meet and join properties follow coordinatewise from Proposition 11. The distributive identities also hold coordinatewise. □

The scalar operations $\oplus$ and $\odot$ extend naturally to vectors and matrices by componentwise application.

Definition 133 (Componentwise MinMax operations)**.** Let $x, y \in \mathbb{R}^n$. Their *componentwise MinMax operations* are defined by

$$(x \oplus y)_i := (x)_i \oplus (y)_i = \max\{(x)_i, (y)_i\},$$
$$(x \odot y)_i := (x)_i \odot (y)_i = \min\{(x)_i, (y)_i\}$$

for every $i \in [1..n]$.

Analogously, for $A, B \in \mathbb{R}^{m\times n}$, we define

$$(A \oplus B)_{i,j} := (A)_{i,j} \oplus (B)_{i,j},$$
$$(A \odot B)_{i,j} := (A)_{i,j} \odot (B)_{i,j}$$

for every $i \in [1..m]$ and $j \in [1..n]$. *End of Definition 133.*

Fact 37 (Componentwise lattice operations)**.** *For every $x, y \in \mathbb{R}^n$,*

$$x \odot y = x \wedge y, \qquad x \oplus y = x \vee y,$$

where the meet and join are taken with respect to $\leq_{\mathrm{cw}}$. The analogous identities hold entrywise for matrices of equal dimensions. End of Fact 37.

D.3 MinMax Matrix Algebra

The scalar MinMax algebra $\mathbb{M}$ induces vector and matrix products analogously to ordinary matrix algebra: scalar multiplication is replaced by $\odot$, finite scalar addition is replaced by $\oplus$, and the resulting matrix product is denoted by $\otimes$.

The use of the word *algebra* in the title of this subsection is descriptive. Because multiplication of rectangular matrices changes dimensions, the collection of all rectangular matrices is not itself a single algebra in the sense of Definition 55.

Definition 134 (MinMax matrix products)**.** Let

$$A \in \mathbb{R}^{m\times n}, \qquad x \in \mathbb{R}^n.$$

Their *MinMax matrix–vector product* is the vector

$$A \otimes x \in \mathbb{R}^m$$

defined by

$$(A \otimes x)_i := \bigoplus_{j=1}^{n} ((A)_{i,j} \odot (x)_j)$$
$$= \max_{j\in[1..n]} \min\{(A)_{i,j}, (x)_j\}$$

for every $i \in [1..m]$.

Let now

$$A \in \mathbb{R}^{m\times n}, \qquad B \in \mathbb{R}^{n\times p}.$$

Their *MinMax matrix product* is the matrix

$$A \otimes B \in \mathbb{R}^{m\times p}$$

defined by

$$(A \otimes B)_{i,j} := \bigoplus_{k=1}^{n} ((A)_{i,k} \odot (B)_{k,j})$$
$$= \max_{k\in[1..n]} \min\{(A)_{i,k}, (B)_{k,j}\}$$

for every $i \in [1..m]$ and $j \in [1..p]$. *End of Definition 134.*

Proposition 15 (MinMax matrix identities)**.** *Whenever the expressions below are dimensionally well-defined, the MinMax matrix operations satisfy*

$$(A \otimes B) \otimes C = A \otimes (B \otimes C), \tag{15}$$
$$(A \otimes B) \otimes x = A \otimes (B \otimes x), \tag{16}$$
$$A \otimes (x \oplus y) = (A \otimes x) \oplus (A \otimes y), \tag{17}$$
$$A \otimes (B \oplus C) = (A \otimes B) \oplus (A \otimes C), \tag{18}$$
$$(A \oplus B) \otimes C = (A \otimes C) \oplus (B \otimes C). \tag{19}$$

Proof. The claims follow from associativity and commutativity of the scalar operations $\oplus$ and $\odot$, together with distributivity of $\odot$ over $\oplus$, which hold by Proposition 12.

We first prove matrix associativity. Let

$$A \in \mathbb{R}^{m \times n}, \qquad B \in \mathbb{R}^{n \times p}, \qquad C \in \mathbb{R}^{p \times q}.$$

For every $i \in [1..m]$ and $j \in [1..q]$,

$$\begin{aligned}
((A \otimes B) \otimes C)_{i,j} &= \bigoplus_{k=1}^{p} \left[\left(\bigoplus_{\ell=1}^{n} \big((A)_{i,\ell} \odot (B)_{\ell,k} \big) \right) \odot (C)_{k,j} \right] \\
&= \bigoplus_{k=1}^{p} \bigoplus_{\ell=1}^{n} \big((A)_{i,\ell} \odot (B)_{\ell,k} \odot (C)_{k,j} \big) \\
&= \bigoplus_{\ell=1}^{n} \bigoplus_{k=1}^{p} \big((A)_{i,\ell} \odot (B)_{\ell,k} \odot (C)_{k,j} \big) \\
&= \bigoplus_{\ell=1}^{n} \left[(A)_{i,\ell} \odot \left(\bigoplus_{k=1}^{p} \big((B)_{\ell,k} \odot (C)_{k,j} \big) \right) \right] \\
&= (A \otimes (B \otimes C))_{i,j}.
\end{aligned}$$

This proves (15).

The matrix–vector associativity (16) follows by the same argument.

For (17), let

$$A \in \mathbb{R}^{m \times n}, \qquad x, y \in \mathbb{R}^{n}.$$

Then, for every $i \in [1..m]$,

$$\begin{aligned}
(A \otimes (x \oplus y))_i &= \bigoplus_{j=1}^{n} \big((A)_{i,j} \odot \big((x)_j \oplus (y)_j \big) \big) \\
&= \bigoplus_{j=1}^{n} \big[\big((A)_{i,j} \odot (x)_j \big) \oplus \big((A)_{i,j} \odot (y)_j \big) \big] \\
&= ((A \otimes x) \oplus (A \otimes y))_i.
\end{aligned}$$

The two matrix distributivity identities (18) and (19) follow entrywise by the same reasoning. □

D.4 MinMax Affine Operators

The MinMax matrix operations induce a class of affine-like maps that is closed under composition.

Definition 135 (MinMax affine operators)**.** Given

$$A \in \mathbb{R}^{n \times n}, \qquad b \in \mathbb{R}^{n},$$

the pair

$$M := \langle A, b \rangle$$

defines the *MinMax affine operator*

$$\mathcal{A}_M : \mathbb{R}^n \to \mathbb{R}^n, \qquad x \mapsto (A \otimes x) \oplus b.$$

Given two MinMax affine operators

$$M_1 = \langle A_1, b_1 \rangle, \qquad M_2 = \langle A_2, b_2 \rangle,$$

we define their *MinMax affine composition* by

$$M_1 \circledast M_2 := \big\langle A_2 \otimes A_1, (A_2 \otimes b_1) \oplus b_2 \big\rangle.$$

End of Definition 135.

Lemma 5 (Composition of MinMax affine operators)**.** *Let $M_1 = \langle A_1, b_1 \rangle$ and $M_2 = \langle A_2, b_2 \rangle$ be MinMax affine operators of the same dimension. Then*

$$\mathcal{A}_{M_1 \circledast M_2} \;=\; \mathcal{A}_{M_2} \circ \mathcal{A}_{M_1}. \tag{20}$$

Furthermore, $\circledast$ is associative:

$$(M_1 \circledast M_2) \circledast M_3 \;=\; M_1 \circledast (M_2 \circledast M_3) \tag{21}$$

for all MinMax affine operators M_1, M_2, M_3 of the same dimension.

Proof. For every $x \in \mathbb{R}^n$, Proposition 15 gives

$$\begin{aligned}
\mathcal{A}_{M_2}\big(\mathcal{A}_{M_1}(x)\big) &= A_2 \otimes \big((A_1 \otimes x) \oplus b_1\big) \oplus b_2 \\
&= \big(A_2 \otimes (A_1 \otimes x)\big) \oplus (A_2 \otimes b_1) \oplus b_2 \\
&= \big((A_2 \otimes A_1) \otimes x\big) \oplus \big((A_2 \otimes b_1) \oplus b_2\big) \\
&= \mathcal{A}_{M_1 \circledast M_2}(x).
\end{aligned}$$

This proves (20).

We prove associativity directly at the level of the representing pairs. Let additionally

$$M_3 = \langle A_3, b_3 \rangle.$$

By Definition 211,

$$\begin{aligned}
(M_1 \circledast M_2) \circledast M_3 &= \big\langle A_3 \otimes (A_2 \otimes A_1),\; A_3 \otimes \big((A_2 \otimes b_1) \oplus b_2\big) \oplus b_3 \big\rangle \\
&= \big\langle (A_3 \otimes A_2) \otimes A_1,\; \big((A_3 \otimes A_2) \otimes b_1\big) \oplus (A_3 \otimes b_2) \oplus b_3 \big\rangle,
\end{aligned}$$

where we used Proposition 15.

On the other hand,

$$\begin{aligned}
M_1 \circledast (M_2 \circledast M_3) &= \big\langle (A_3 \otimes A_2) \otimes A_1,\; \big((A_3 \otimes A_2) \otimes b_1\big) \oplus \big((A_3 \otimes b_2) \oplus b_3\big) \big\rangle \\
&= \big\langle (A_3 \otimes A_2) \otimes A_1,\; \big((A_3 \otimes A_2) \otimes b_1\big) \oplus (A_3 \otimes b_2) \oplus b_3 \big\rangle,
\end{aligned}$$

by associativity of $\oplus$. Hence (21) follows. □

Remark 17. By associativity, finite compositions of MinMax affine operators are unambiguously written as

$$M_1 \circledast \cdots \circledast M_T.$$

End of Remark 17.

D.5 MinMax Recurrences

A MinMax recurrence is obtained by applying a sequence of MinMax affine operators to a recurrent state.

Proposition 16 (Finite-horizon MinMax recurrence)**.** *Let $x_0 \in \mathbb{R}^n$, let $T \in \mathbb{N}^{>0}$, let*

$$(R_t)_{t=1}^T, \qquad (s_t)_{t=1}^T, \qquad (x_t)_{t=1}^T, \qquad (M_t)_{t=1}^T$$

be as follows, for every $t \in [1..T]$:

$$R_t \in \mathbb{R}^{n \times n}, \qquad s_t \in \mathbb{R}^n, \qquad x_t := \big(R_t \otimes x_{t-1}\big) \oplus s_t, \qquad M_t := \langle R_t, s_t \rangle.$$

Then,

$$x_t = \mathcal{A}_{M_t}(x_{t-1}) \qquad \forall t \in [1..T],$$

and $x_T = \mathcal{A}_{M_1 \circledast \cdots \circledast M_T}(x_0)$.

Proof. The one-step identity follows directly from Definition 211.

Repeated application of Lemma 5 gives

$$x_T \;=\; \mathcal{A}_{M_T} \circ \cdots \circ \mathcal{A}_{M_1}(x_0) \;=\; \mathcal{A}_{M_1 \circledast \cdots \circledast M_T}(x_0),$$

where the parenthesisation is immaterial by associativity of $\circledast$. □

Corollary 9 (Finite MinMax-affine representation)**.** *Under the assumptions of Proposition 16, there exist*

$$A_T \in \mathbb{R}^{n \times n}, \qquad b_T \in \mathbb{R}^n$$

such that

$$M_1 \circledast \cdots \circledast M_T = \langle A_T, b_T \rangle,$$

and consequently

$$x_T \;=\; (A_T \otimes x_0) \oplus b_T \,. \tag{22}$$

Proof. The class of MinMax affine operators is closed under $\circledast$ by Definition 211. Hence their finite composition has the form $\langle A_T, b_T \rangle$. The result then follows from Proposition 16. □

D.6 Finite Restrictions of the MinMax Algebra

Moving from $\mathbb{R}$ to its extension $\mathbb{R} \cup \{+\infty, -\infty\}$, one can extend the scalar MinMax algebra over $\mathbb{R}$ to the MinMax semiring over $\mathbb{R} \cup \{+\infty, -\infty\}$,

$$(\mathbb{R} \cup \{+\infty, -\infty\},\, \oplus,\, \odot,\, +\infty,\, -\infty),$$

where $+\infty$ is the identity element for $\oplus$ and $-\infty$ is the identity element for $\odot$.

However, in this work we do not adopt such extension, as it seems unnecessary. In fact, for finite computations, the relevant MinMax algebra is the one over the finite domain V of values that occur during a computation. This restricted variant of the MinMax algebra retains the key algebraic properties of the one over $\mathbb{R}$, and additionally it admits identity elements for both $\oplus$ and $\odot$, which are the smallest and largest values of V.

Definition 136 (Finite MinMax restriction)**.** Let $V \subseteq \mathbb{R}$ be a nonempty finite set, and define

$$v_{\min} := \min V, \qquad v_{\max} := \max V.$$

The *finite MinMax restriction induced by* V is $(V, \oplus, \odot)$ where $\oplus$ and $\odot$ are the restrictions to V of the scalar MinMax operations from Definition 131. *End of Definition 136.*

Observe that every subset of a totally ordered set is closed under binary minimum and maximum. Hence the above operations are well-defined on V.

Proposition 17 (Identities of a finite MinMax restriction)**.** *Let $V \subseteq \mathbb{R}$ be nonempty and finite. Then, for every $x \in V$,*

$$v_{\min} \oplus x \;=\; x \oplus v_{\min} \;=\; x \qquad \textit{and} \qquad v_{\max} \odot x \;=\; x \odot v_{\max} \;=\; x.$$

Furthermore, for every $x \in V$,

$$v_{\min} \odot x \;=\; x \odot v_{\min} \;=\; v_{\min}.$$

Consequently, we have that $(V, \oplus, \odot, v_{\min}, v_{\max})$ is a finite commutative idempotent semiring, with additive identity $v_{\min}$ and multiplicative identity $v_{\max}$.

Proof. Since $v_{\min} \le x \le v_{\max}$,

$$\begin{aligned} v_{\min} \oplus x &= \max\{v_{\min}, x\} = x, \\ v_{\max} \odot x &= \min\{v_{\max}, x\} = x, \\ v_{\min} \odot x &= \min\{v_{\min}, x\} = v_{\min}. \end{aligned}$$

Commutativity gives the corresponding right identities. The remaining semiring properties follow from Fact 36. □

The finite restrictions above arise naturally when evaluating a MinMax recurrence over a finite input sequence.

Definition 137 (Data-generated MinMax domain)**.** Let $T, n \in \mathbb{N}^{>0}$, let $x_0 \in \mathbb{R}^n$, and let $(R_t)_{t=1}^T$ and $(s_t)_{t=1}^T$ be sequence of functions

$$R_t \in \mathbb{R}^{n\times n}, \qquad s_t \in \mathbb{R}^n.$$

The *MinMax domain generated by the data* $\big(x_0, (R_t, s_t)_{t=1}^T\big)$ is the finite set

$$V_{\circledast,} := \operatorname{entries}(x_0) \cup \bigcup_{t=1}^{T} \operatorname{entries}(R_t) \cup \bigcup_{t=1}^{T} \operatorname{entries}(s_t)\ .$$

We write

$$v_{\min} := \min V_{\circledast,}, \qquad v_{\max} := \max V_{\circledast,}\ .$$

End of Definition 137.

Since $\oplus$ and$\odot$ ar selective operations—that is, each returns one of its arguments—MinMax computations cannot generate scalar values outside the data-generated domain.

Proposition 18 (Closure of finite MinMax computations)**.** *Let $V_{\circledast,}$ be as in Definition 137. Then:*

- *if $A \in V_{\circledast,}^{m\times n}$ and $x \in V_{\circledast,}^{n}$, then $A \otimes x \in V_{\circledast,}^{m}$;*
- *if $A_1 \in V_{\circledast,}^{m\times n}$ and $A_2 \in V_{\circledast,}^{n\times p}$, then $A_1 \otimes A_2 \in V_{\circledast,}^{m\times p}$;*
- *if $M_1 = \langle A_1, b_1\rangle$ and $M_2 = \langle A_2, b_2\rangle$ have all their entries in $V_{\circledast,}$, then $M_1 \circledast M_2$ has all its entries in $V_{\circledast,}$;*
- *every state x_t of the recurrence $x_t = (R_t \otimes x_{t-1}) \oplus s_t$ belongs to $V_{\circledast,}^{n}$.*

Proof. Every application of $\oplus$ or $\odot$ to elements of $V_{\circledast,}$ returns one of its arguments, and hence again an element of $V_{\circledast,}$. The first two claims therefore follow directly from Definition 134. The third follows from Definition 211. The last claim follows by induction on t, starting from $x_0 \in V_{\circledast,}^{n}$. □

We can therefore construct an exact identity for MinMax affine composition using only scalar values appearing in the data.

Proposition 19 (Data-generated identity for MinMax affine composition)**.** *Let $V_{\circledast,}$ be as in Definition 137, and let*

$$E \in V_{\circledast,}^{n\times n}, \qquad e \in V_{\circledast,}^{n},$$

be defined by

$$(E)_{i,j} := \begin{cases} v_{\max} & \text{if } i = j, \\ v_{\min} & \text{if } i \neq j, \end{cases} \qquad (e)_i := v_{\min}.$$

Equivalently,

$$E = v_{\min} \cdot \mathbf{1}_{n,n} + (v_{\max} - v_{\min}) \cdot \mathbf{I}_{n,n}, \qquad e = v_{\min} \cdot \mathbf{1}_n.$$

Then, for every $M = \langle A, b\rangle \in V_{\circledast,}^{n\times n} \times V_{\circledast,}^{n}$, we have

$$\langle E, e\rangle \circledast M \;=\; M \circledast \langle E, e\rangle \;=\; M.$$

Hence, $\big(V_{\circledast,}^{n\times n} \times V_{\circledast,}^{n}, \circledast, \langle E, e\rangle\big)$ is a finite monoid.

Proof. By Proposition 17, $v_{\min}$ is the identity of $\oplus$ and $v_{\max}$ is the identity of $\odot$ over $V_{\circledast,}$. Therefore, for every $A \in V_{\circledast,}^{n\times n}$ and $b \in V_{\circledast,}^{n}$,

$$E \otimes A = A, \qquad A \otimes E = A,$$
$$E \otimes b = b, \qquad A \otimes e = e.$$

Hence, by Definition 211,

$$\langle E, e\rangle \circledast \langle A, b\rangle = \langle A \otimes E, (A \otimes e) \oplus b\rangle = \langle A, b\rangle,$$
$$\langle A, b\rangle \circledast \langle E, e\rangle = \big\langle E \otimes A,\ (E \otimes b) \oplus e\big\rangle = \langle A, b\rangle.$$

Associativity of $\circledast$ follows from Lemma 5. Finiteness follows from finiteness of $V_{\circledast,}$. □

Proposition 20 (Data-generated MinMax transformation monoid)**.** *Let*

$$\mathcal{M}_{\circledast} := V_{\circledast,}^{n\times n} \times V_{\circledast,}^{n}$$

be the finite monoid of MinMax affine operators from Proposition 19, with operation $\circledast$ and identity

$$E_{\circledast} := \langle E, e\rangle.$$

Define the set of induced transformations

$$\mathcal{T}_{\circledast} := \left\{ \mathcal{A}_M|_{V_{\circledast,}^{n}} \mid M \in \mathcal{M}_{\circledast} \right\}.$$

Then every

$$\tau \in \mathcal{T}_{\circledast}$$

is a transformation

$$\tau : V_{\circledast,}^{n} \to V_{\circledast,}^{n},$$

and

$$(V_{\circledast,}^{n}, \mathcal{T}_{\circledast})$$

is a finite transformation monoid under standard function composition.

Furthermore, the action map

$$\rho : \mathcal{M}_{\circledast} \to \mathcal{T}_{\circledast}, \qquad M \mapsto \mathcal{A}_M|_{V_{\circledast,}^{n}}$$

is surjective and satisfies

$$\rho(M_1 \circledast M_2) = \rho(M_2) \circ \rho(M_1). \tag{23}$$

Thus, ρ is an anti-homomorphism from $\mathcal{M}_{\circledast}$ onto the induced transformation monoid, or equivalently a homomorphism from the opposite monoid $\mathcal{M}_{\circledast}^{\mathrm{op}}$.

Proof. By Proposition 18, every $M = \langle A, b\rangle \in \mathcal{M}_{\circledast}$ induces a well-defined transformation

$$\mathcal{A}_M : V_{\circledast,}^{n} \to V_{\circledast,}^{n}.$$

By Lemma 5, for every $M_1, M_2 \in \mathcal{M}_{\circledast}$,

$$\mathcal{A}_{M_1 \circledast M_2} = \mathcal{A}_{M_2} \circ \mathcal{A}_{M_1}.$$

Hence $\mathcal{T}_{\circledast}$ is closed under function composition: if $\tau_1 = \mathcal{A}_{M_1}|_{V_{\circledast,}^{n}}$ and $\tau_2 = \mathcal{A}_{M_2}|_{V_{\circledast,}^{n}}$, then

$$\tau_2 \circ \tau_1 = \mathcal{A}_{M_1 \circledast M_2}|_{V_{\circledast,}^{n}} \in \mathcal{T}_{\circledast}.$$

Proposition 19 gives

$$\mathcal{A}_{E_{\circledast}}|_{V_{\circledast,}^{n}} = \mathrm{id}_{V_{\circledast,}^{n}}.$$

so the identity transformation belongs to $\mathcal{T}_{\circledast}$. Associativity follows from associativity of standard function composition. Finiteness follows from finiteness of $\mathcal{M}_{\circledast}$.

Finally, surjectivity of ρ follows directly from the definition of $\mathcal{T}_{\circledast}$, while Equation (23) is precisely Lemma 5. □

Corollary 10 (Prefix products represent recurrence transformations)**.** *Under the assumptions of Proposition 16, let*

$$\tau_t := \mathcal{A}_{M_t}|_{V_{\circledast,}^{n}}.$$

Then, for every $t \in [1..T]$,

$$\mathcal{A}_{M_1 \circledast \cdots \circledast M_t}|_{V_{\circledast,}^{n}} = \tau_t \circ \cdots \circ \tau_1, \tag{24}$$

and consequently

$$x_t = \mathcal{A}_{M_1 \circledast \cdots \circledast M_t}(x_0) = (\tau_t \circ \cdots \circ \tau_1)(x_0).$$

Proof. Repeated application of Equation (23) gives

$$\rho\big(M_1 \circledast \cdots \circledast M_t\big) = \rho(M_t) \circ \cdots \circ \rho(M_1).$$

By the definitions of ρ and τ_t, this is precisely Equation (24). The state identity then follows from Proposition 16. □

Thus, the parallel algorithms do not require extending the scalar domain to $\mathbb{R} \cup \{-\infty, +\infty\}$. For every finite input sequence, they operate exactly over the finite MinMax restriction generated by the values occurring in that computation. In particular, $v_{\min}$ and $v_{\max}$ do not merely approximate $-\infty$ and $+\infty$: within the data-generated domain they are respectively the exact identities of $\oplus$ and $\odot$. The identity element required by parallel scan is therefore derived entirely from the finite data of the computation.

E Selected Intensional-Derivative Operators

Table of Contents

E.1 Computation Graphs 89
E.2 Pseudo-derivative Operators 100
E.3 Selected Pseudo-Derivative Operators 104
E.4 Intensional-Derivative Operators 105
E.5 Graph-Induced Intensional-Derivative Operators 105
E.6 Selected Intensional-Derivative Operators 108
E.7 Analytic Extensions of Computation Graphs 110

E.1 Computation Graphs

Definition 138 (Computation graph)**.** Let $\mathcal{P}$ be a class of functions.

A *$\mathcal{P}$-computation graph*, or simply *computation graph*, G of input dimension $d_{\text{in}} \in \mathbb{N}$ and output dimension $d_{\text{out}} \in \mathbb{N}^{>0}$, is a tuple

$$G = \langle V, E, \mathring{v}^{\text{in}}, \mathring{v}^{\text{c}}, \mathring{v}^{\text{out}}, \lambda \rangle$$

consisting of the following components:

- a finite set nodes V;
- the *input node-vector*

  $$\mathring{v}^{\text{in}} = (\mathring{v}_1^{\text{in}}, \dots, \mathring{v}_{d_{\text{in}}}^{\text{in}})$$

  consisting of of pair-wise distinct nodes from V, that we call *input nodes*;
- the *output node-vector*

  $$\mathring{v}^{\text{out}} = (\mathring{v}_1^{\text{out}}, \dots, \mathring{v}_{d_{\text{out}}}^{\text{out}})$$

  consisting of of pair-wise distinct nodes from $V \in \text{entries}\big(\mathring{v}^{\text{in}}\big)$, that we call *output nodes*;
- the *computation node-vector* $\mathring{v}^{\text{c}}$ consisting of pair-wise distinct nodes from $V \setminus \text{entries}\big(\mathring{v}^{\text{in}}\big)$, that we call *computation nodes*, and such that $\text{entries}\big(\mathring{v}^{\text{in}}\big) \cup \text{entries}(\mathring{v}^{\text{c}}) = V$;
- a finite set of directed multi-edges

  $$E \subseteq V^* \times V_{\text{c}},$$

  such that:
  - every computation node has exactly one incoming multi-edge;
  - the resulting directed graph is acyclic;

  if the unique incoming multi-edge of a computation node $\mathring{v}$ is

  $$(\mathring{u}_1, \dots, \mathring{u}_k) \to \mathring{v},$$

  then k is the *arity* of $\mathring{v}$, and

  $$\mathring{u} := (\mathring{u}_1, \dots, \mathring{u}_k)$$

  is its *predecessor node-vector*,
- a node-labelling function $\lambda : \text{entries}(V_{\text{c}}) \to \mathcal{P}$ such that, for every computation node $\mathring{v} \in \text{entries}(V_{\text{c}})$, its arity k matches the input-arity of its label $\lambda(\mathring{v}) : \mathbb{R}^k \to \mathbb{R}$.

For every node $\mathring{v} \in V$, we write $\check{v} := \lambda(\mathring{v})$ when the labelling function λ is clear from the context.

The *value function* of the nodes of G is defined inductively as follows:

- for every $i \in [1..d_{\mathrm{in}}]$, the value of the input node $\mathring{v}_i^{\mathrm{in}}$ is

$$\hat{v}_i^{\mathrm{in}} : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}, \qquad \hat{v}_i^{\mathrm{in}}(x) := (x)_i;$$

- for every computation node $\mathring{v} \in V_{\mathrm{c}}$, having predecessor node-vector

$$\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k)$$

its value function is defined inductively by

$$\hat{v}(x) := \check{v}\big(\hat{u}_i(x), \ldots, \hat{u}_k(x)\big).$$

Then, the *value function of the graph G* is

$$\hat{G} : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}}, \qquad \hat{G}(x) := \big(\hat{v}_1^{\mathrm{out}}(x), \ldots, \hat{v}_{d_{\mathrm{out}}}^{\mathrm{out}}(x)\big).$$

We say that G *represents* the function $\hat{G}$. *End of Definition 138.*

Definition 139 (Computation graph with virtual computation nodes)**.** Let $\mathcal{P}$ be a class of functions, let $d_{\mathrm{in}} \in \mathbb{N}$, and let $d_{\mathrm{out}} \in \mathbb{N}^{>0}$.

A *$\mathcal{P}$-computation graph with virtual computation nodes*, also called a *$\mathcal{P}$-virtual-computation graph* or simply a *virtual-computation graph*, is a tuple

$$G := \langle V, V_{\mathrm{v}}, E, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \rangle$$

consisting of the following components:

- a finite set V of *concrete nodes*;
- a finite set V_{v} of *virtual computation nodes* satisfying $V \cap V_{\mathrm{v}} = \emptyset$;
- the *input node-vector* $\mathring{v}^{\mathrm{in}}$ consisting of pair-wise distinct concrete nodes from V, called the *input nodes*;
- the *concrete computation node-vector* $\mathring{v}^{\mathrm{c}}$ consisting of d_{in} pair-wise distinct concrete nodes from $V \setminus \mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big)$, called the *concrete computation nodes*, and satisfying $\mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big) \cup \mathrm{entries}(\mathring{v}^{\mathrm{c}}) = V$;
- the *output node-vector* $\mathring{v}^{\mathrm{out}}$ consisting of d_{out} pair-wise distinct nodes from $V \cup V_{\mathrm{v}}$, called the *output nodes*;
- a finite set of directed multi-edges

$$E \subseteq (V \cup V_{\mathrm{v}})^* \times (\mathrm{entries}(\mathring{v}^{\mathrm{c}}) \cup V_{\mathrm{v}}),$$

satisfying the following conditions:

  - every concrete computation node and every virtual computation node has exactly one incoming multi-edge;
  - the directed graph induced by E over $V \cup V_{\mathrm{v}}$ is acyclic;

for $\mathring{u} \to \mathring{v}$ the unique incoming multi-edge of a concrete or virtual computation node $\mathring{v}$, the *(input) arity* of $\mathring{v}$, is $\mathrm{ar}_{\mathrm{in}}(\mathring{v}) := |\mathring{u}|$ and its *predecessor node-vector* is $\mathring{u}$;

- a node-labelling function

$$\lambda : \mathrm{entries}(\mathring{v}^{\mathrm{c}}) \cup V_{\mathrm{v}} \to \mathcal{P}$$

such that the arity of every concrete or virtual computation node $\mathring{v}$ matches the input arity of its label, namely $\mathrm{ar}_{\mathrm{in}}(\mathring{v}) = \mathrm{ar}_{\mathrm{in}}(\lambda(\mathring{v}))$.

We call $\mathrm{ar}_{\mathrm{in}}(G) := d_{\mathrm{in}}$ the *input dimension* or *input arity* of G, and we call $\mathrm{ar}_{\mathrm{out}}(G) := d_{\mathrm{out}}$ the *output dimension* or *output arity* of G.

When no ambiguity arises, we call a virtual computation node simply a *virtual node*.

A *node* of G is a concrete node or a virtual node of G. A *computation node* of G is a concrete computation node or a virtual computation node of G.

For every computation node $\mathring{v}$, we write

$$\check{v} := \lambda(\mathring{v})$$

when the labelling function λ is clear from the context.

The *value functions* of the concrete and virtual nodes of G are defined inductively as follows:

- for every $i \in [1..d_{\text{in}}]$, the value function of the input node $\mathring{v}_i^{\text{in}}$ is

$$\hat{v}_i^{\text{in}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}, \qquad \hat{v}_i^{\text{in}}(x) := (x)_i;$$

- for every computation node $\mathring{v}$ having predecessor node-vector $\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k)$, its value function is defined inductively by

$$\hat{v}(x) := \check{v}\big(\hat{u}_1(x), \ldots, \hat{u}_k(x)\big).$$

The *value function of G* is

$$\hat{G} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}, \qquad \hat{G}(x) := \big(\hat{v}_1^{\text{out}}(x), \ldots, \hat{v}_{d_{\text{out}}}^{\text{out}}(x)\big).$$

We say that G *represents* the function $\hat{G}$.

A computation graph in the sense of Definition 138 is identified with a virtual-computation graph where $V_{\text{v}} = \emptyset$. *End of Definition 139.*

Definition 140 (Virtual-node extension)**.** Let

$$G = \langle V, V_{\text{v}}, E, \mathring{v}^{\text{in}}, \mathring{v}^{\text{c}}, \mathring{v}^{\text{out}}, \lambda \rangle$$

be a virtual-computation graph, let $\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k) \in (V \cup V_{\text{v}})^k$ be a node-vector presentation of G, let $f : \mathbb{R}^k \to \mathbb{R}$ be a scalar Euclidean function, and let $\mathring{w} \notin V \cup V_{\text{v}}$ be a fresh node.

The *virtual-node extension of G obtained by adjoining $\mathring{w}$ with predecessor node-vector $\mathring{u}$ and label f* is the virtual-computation graph

$$G' := \langle V, V'_{\text{v}}, E', \mathring{v}^{\text{in}}, \mathring{v}^{\text{c}}, \mathring{v}^{\text{out}}, \lambda' \rangle$$

where

$$V'_{\text{v}} := V_{\text{v}} \cup \{\mathring{w}\}, \qquad E' := E \cup \{\mathring{u} \to \mathring{w}\}, \qquad \lambda' := \lambda \cup \{\mathring{w} \mapsto f\}.$$

We say that $\mathring{w}$ is the *virtual node adjoined by the extension*.

A *virtual-node extension* of G is any virtual-computation graph obtained from G by finitely many successive applications of this construction. *End of Definition 140.*

Definition 141 (Materialisation)**.** Let V_{fresh} be an infinite ordered set of nodes satisfying $V_{\text{fresh}} \cap V = \emptyset$, and let G be a virtual-computation graph given by

$$G = \langle V, V_{\text{v}}, E, \mathring{v}^{\text{in}}, \mathring{v}^{\text{c}}, \mathring{v}^{\text{out}}, \lambda \rangle.$$

A *materialisation map* for G with fresh nodes from V_{fresh} is an injective map $\mu : V_{\text{v}} \to V_{\text{fresh}}$. We extend μ to all concrete and virtual nodes by defining

$$\bar{\mu}(\mathring{v}) := \begin{cases} \mathring{v} & \text{if } \mathring{v} \in V, \\ \mu(\mathring{v}) & \text{if } \mathring{v} \in V_{\text{v}}, \end{cases}$$

and we apply $\bar{\mu}$ components-wise to node-vector and node-matrix presentations.

The *materialisation of G under μ* is the computation graph

$$G^{\mu} := \langle V^{\mu}, E^{\mu}, \mathring{v}^{\text{in},\mu}, \mathring{v}^{\text{c},\mu}, \mathring{v}^{\text{out},\mu}, \lambda^{\mu} \rangle$$

having components as follows:

- the set of all nodes, the input node-vector, and the output node-vector are as follows:

$$V^{\mu} := V \cup \mu(V_{\text{v}}) \qquad \mathring{v}^{\text{in},\mu} := \mathring{v}^{\text{in}}, \qquad \mathring{v}^{\text{out},\mu} := \bar{\mu}\big(\mathring{v}^{\text{out}}\big);$$

- the computation node-vector $\mathring{v}^{\mathrm{c},\mu}$ is an ordering of the set $\mathrm{entries}(\mathring{v}^{\mathrm{c}}) \cup \mu(V_{\mathrm{v}})$ according to the order provided by V_{fresh};
- the multi-edge set is
$$E^{\mu} := \big\{ \bar{\mu}(\mathring{u}) \to \bar{\mu}(\mathring{v}) \;\big|\; \mathring{u} \to \mathring{v} \in E \big\},$$
where $\bar{\mu}$ is applied entrywise to predecessor node-vectors;
- finally, the labelling function is given by
$$\lambda^{\mu}(\mathring{v}) := \lambda\big(\bar{\mu}^{-1}(\mathring{v})\big) \qquad \forall \mathring{v} \in \mathrm{entries}(\mathring{v}^{\mathrm{c},\mu})\,.$$

We call G^{μ} a *materialised graph of* G. A concrete or virtual node, node-vector, or node-matrix presentation is *materialised* by applying $\bar{\mu}$ entrywise. *End of Definition 141.*

Proposition 21 (Correctness of materialisation)**.** *Let G be a $\mathcal{P}$-computation graph with virtual computation nodes, and let $G^{\mu} := G^{\mu}$ be a materialisation of G.*

Then:

- *G^{μ} is a $\mathcal{P}$-computation graph in the sense of Definition 138;*
- *materialisation preserves the value function of every node: if $\mathring{v}^{\mu} := \bar{\mu}(\mathring{v})$, then $\hat{v^{\mu}} = \hat{v}$;*
- *consequently, materialisation preserves the value function of every node-vector and node-matrix presentation;*
- *in particular, $\hat{G^{\mu}} = \hat{G}$.*

Hence G and every materialised graph of G represent the same function. End of Proposition 21.

Proof. Since μ is injective and its image is disjoint from V, the map $\bar{\mu}$ is injective. Hence pair-wise distinct nodes remain pair-wise distinct under materialisation.

Every concrete computation node retains its original predecessors, while every virtual computation node is replaced by a concrete computation node whose predecessors are the materialisations of its original predecessors. Thus every computation node of G^{μ} has exactly one incoming multi-edge. Moreover, injective relabelling of the nodes preserves acyclicity. Therefore G^{μ} is a computation graph.

Since materialisation preserves every node label, and every concrete or virtual computation-node label of G belongs to $\mathcal{P}$, the graph G^{μ} is a $\mathcal{P}$-computation graph.

We prove preservation of node-value functions by induction along a topological ordering of G. The input nodes are unchanged, so the claim holds for them.

Let $\mathring{v}$ be a concrete or virtual computation node with predecessor node-vector
$$\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k),$$
and suppose that the claim holds for its predecessors. Write
$$\mathring{v}^{\mu} := \bar{\mu}(\mathring{v}), \qquad \mathring{u}_i^{\mu} := \bar{\mu}(\mathring{u}_i).$$
Then
$$\begin{aligned} \hat{v^{\mu}}(x) &= \check{v}\big(u[\hat{1}]^{\mu}(x), \ldots, u[\hat{k}]^{\mu}(x)\big) \\ &= \check{v}\big(\hat{u}_1(x), \ldots, \hat{u}_k(x)\big) \\ &= \hat{v}(x). \end{aligned}$$
This proves preservation of all node-value functions.

The claim for node-vector and node-matrix presentations follows componentwise. Applying it to the output node-vector gives
$$\hat{G^{\mu}} = \hat{G},$$
as required. □

Fact 38 (Uniqueness of materialisation)**.** *Any two materialisations of the same computation graph with virtual computation nodes differ only by the names of the fresh computation nodes and, possibly, by the ordering chosen for the computation node-vector.*

In particular, materialisation is unique up to renaming of the materialised virtual nodes and permutation of the computation node-vector. End of Fact 38.

Definition 142 (Node-vector and node-matrix presentations)**.** Let us consider a computation graph

$$G = \langle V, E, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \rangle.$$

Then,

- a *node-vector presentation* for nodes of G, having dimension $m \in \mathbb{N}$ is a vector $\mathring{v} \in V^m$, and its value function is
$$\hat{v} : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^m, \qquad \hat{v}(x) := (\hat{v}_1(x), \ldots, \hat{v}_m(x));$$
- a *node-matrix presentation* for nodes of G, having dimension $m \times n$ for $m, n \in \mathbb{N}^{>0}$, is a matrix $\mathring{A} :\in V^{m \times n}$, and its value function is
$$\hat{A} : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{m \times n}, \qquad \hat{A}(x) := \big((\hat{A})_{i,j}(x)\big)_{i \in [1..m],\, j \in [1..n]}.$$

Node-vector and node-matrix presentations are structured collections of scalar nodes and do not constitute additional kinds of nodes.

In particular, the input node-vector $\mathring{v}^{\mathrm{in}}$, every predecessor node-vector $\mathring{u}$, and the output node-vector $\mathring{v}^{\mathrm{out}}$ of a computation graph are node-vector presentations. *End of Definition 142.*

Definition 143 (Node presentations with virtual nodes)**.** Let

$$G = \langle V, V_{\mathrm{v}}, E, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \rangle$$

be a computation graph with virtual computation nodes.

Node-vector presentations and *node-matrix presentations* of G are defined as in Definition 142, with entries taken from $V \cup V_{\mathrm{v}}$. Also their value functions are as in Definition 142.

A node-vector or node-matrix presentation is *concrete* if all of its entries belong to V, and *virtual* otherwise. *End of Definition 143.*

Definition 144 (Independence from graph nodes)**.** Let G be a computation graph, and let $\mathring{u}$ be a node-vector presentation of G.

A node $\mathring{v}$ of G is *independent of* $\mathring{u}$ if it is not reachable by a directed path from any node $(\mathring{u})_i$ for $i \in [1..m]$.

A node-vector or node-matrix presentation is *independent of* $\mathring{u}$ if each of its scalar nodes is independent of $\mathring{u}$.

The computation graph G is *independent of* $\mathring{u}$ if its output node-vector is independent of $\mathring{u}$. *End of Definition 144.*

Definition 145 (Restriction of a computation graph)**.** Let G be a computation graph, let $\mathring{u}$ be a node-vector consisting of distinct input nodes of G, and let $u \in \mathbb{R}^{|\mathring{u}|}$.

The *restriction of G obtained by fixing $\mathring{u} = u$*, denoted by

$$G|_{\mathring{u}=u},$$

is obtained by replacing every $(\mathring{u})_i$ by a zero-arity computation node labelled by the constant $(u)_i$.

Several disjoint input node-vectors may be fixed simultaneously. For instance, for $\mathring{w}$ an additional node-vector consisting of pair-wise distinct input nodes of G and such that $\mathrm{entries}(\mathring{w}) \cap \mathrm{entries}(\mathring{u}) = \emptyset$, we may write

$$G|_{\mathring{u}=u,\, \mathring{w}=w}$$

for $w \in \mathbb{R}^{|\mathring{w}|}$. *End of Definition 145.*

Fact 39 (Correctness of computation-graph restriction)**.** *Restricting a computation graph amounts to partially evaluating its value function at the corresponding fixed input values.*

In particular, for G a computation graph with input node-vector split into two node-vectors as

$$\mathring{v}^{\mathrm{in}} = (\mathring{u}, \mathring{w})$$

letting

$$G_u := G|_{\mathring{u}=u} \qquad \forall u \in \mathbb{R}^{|\mathring{u}|},$$

we have

$$\hat{G}_u(w) = \hat{G}(u, w) \qquad \forall u \in \mathbb{R}^{|\mathring{u}|},\ \forall w \in \mathbb{R}^{|\mathring{w}|}.$$

End of Fact 39.

Definition 146 (Output restriction of a computation graph)**.** Let us consider a computation graph

$$G := \left\langle V, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \right\rangle$$

having input dimension d_{in} and output dimension d_{out}, and let $I := (i_1, \ldots, i_n)$ be a vector of pair-wise distinct indices from $[1..d_{\mathrm{out}}]$, for some $n \in [1..d_{\mathrm{out}}]$.

The *output-projection of G to coordinates I*, is the computation graph

$$G|_I := \left\langle V, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}_I, \lambda \right\rangle$$

$$\text{where } \mathring{v}^{\mathrm{out}}_I := \big((V^{\mathrm{out}})_{i_1}, \ldots, (V^{\mathrm{out}})_{i_n}\big).$$

For $j \in [1..d_{\mathrm{out}}]$, the *scalar output-projection of G to its j-th output coordinate* is the computation graph

$$G|_j := G|_J$$

for J the singleton tuple $J := (j)$. *End of Definition 146.*

Fact 40 (Correctness of computation-graph output projection)**.** *Projecting the output of a computation graph amounts to projecting the corresponding coordinates of its value function, as described next.*

Let G represent

$$f : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}}, \qquad f(x) = \big(f_1(x), \ldots, f_{d_{\mathrm{out}}}(x)\big),$$

and let $I := (i_1, \ldots, i_n)$ be a vector of pair-wise distinct indices from $[1..d_{\mathrm{out}}]$, for some $n \in [1..d_{\mathrm{out}}]$.

Then, the output-projection $G|_I$ represents the function

$$f_I : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^n, \qquad f_I(x) = \big(f_{i_1}(x), \ldots, f_{i_n}(x)\big).$$

Equivalently,

$$\hat{G}|_I(x) = \big(\hat{G}_{i_1}(x), \ldots, \hat{G}_{i_n}(x)\big) \qquad \forall x \in \mathbb{R}^{d_{\mathrm{in}}}.$$

In particular, for every $j \in [1..d_{\mathrm{out}}]$, the scalar output-projection $G|_j$ represents the scalar function

$$f_j : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R},$$

and it satisfies

$$\hat{G}|_j(x) = \hat{G}_j(x) = f_j(x) \qquad \forall x \in \mathbb{R}^{d_{\mathrm{in}}}.$$

End of Fact 40.

Definition 147 (Computation-graph isomorphism)**.** Let us consider a pair of computation graphs

$$G_1 = \langle V_1, E_1, \mathring{v}^{\mathrm{in}}_1, \mathring{v}^{\mathrm{c}}_1, \mathring{v}^{\mathrm{out}}_1, \lambda_1 \rangle, \qquad G_2 = \langle V_2, E_2, \mathring{v}^{\mathrm{in}}_2, \mathring{v}^{\mathrm{c}}_2, \mathring{v}^{\mathrm{out}}_2, \lambda_2 \rangle$$

where each G_i has input and output dimension d^{in}_i and d^{out}_i, respectively.

A *computation-graph isomorphism from G_1 to G_2* is a graph isomorphism

$$\rho : V_1 \to V_2$$

satisfying the following conditions:

- the input and output dimensions agree:
$$d_1^{\mathrm{in}} = d_2^{\mathrm{in}}, \qquad d_1^{\mathrm{out}} = d_2^{\mathrm{out}};$$
- input nodes and their ordering are preserved:
$$\rho\big((\mathring{v}_1^{\mathrm{in}})_i\big) = (\mathring{v}_2^{\mathrm{in}})_i \qquad \forall i \in [1..d_1^{\mathrm{in}}];$$
- computation nodes are preserved:
$$\rho(\mathring{v}_1^{\mathrm{c}}) = \mathring{v}_2^{\mathrm{c}};$$
- output nodes and their ordering are preserved:
$$\rho\big((\mathring{v}_1^{\mathrm{out}})_i\big) = (\mathring{v}_2^{\mathrm{out}})_i \qquad \forall i \in [1..d_1^{\mathrm{out}}];$$
- computation-node labels are preserved:
$$\lambda_2(\rho(\mathring{v})) = \lambda_1(\mathring{v}) \qquad \forall \mathring{v} \in \mathring{v}_1^{\mathrm{c}}.$$

Two computation graphs are called *isomorphic*, or *equal up to a renaming of nodes*, if there exists a computation-graph isomorphism between them. We again denote this relation by
$$G_1 \cong G_2.$$

End of Definition 147.

Proposition 22 (Isomorphic graphs are semantically equivalent)**.** *Let G_1 and G_2 be computation graphs. If*
$$G_1 \cong G_2,$$
then they represent the same function:
$$\hat{G}_1 = \hat{G}_2.$$

Proof. Let
$$G_1 = \langle V_1, E_1, \mathring{v}_1^{\mathrm{in}}, \mathring{v}_1^{\mathrm{c}}, \mathring{v}_1^{\mathrm{out}}, \lambda_1 \rangle, \qquad G_2 = \langle V_2, E_2, \mathring{v}_2^{\mathrm{in}}, \mathring{v}_2^{\mathrm{c}}, \mathring{v}_2^{\mathrm{out}}, \lambda_2 \rangle,$$
and let $\rho : V_1 \to V_2$ be a computation-graph isomorphism from G_1 to G_2. We show, following a topological ordering of the computation nodes of G_1, that every node $\mathring{v} \in V_1$ and its corresponding node $\rho(\mathring{v}) \in V_2$ have the same value function.

The claim holds for the input nodes. In fact, for every $i \in [1..d_1^{\mathrm{in}}]$, Definition 147 gives
$$\rho\big((\mathring{v}_1^{\mathrm{in}})_i\big) = (\mathring{v}_2^{\mathrm{in}})_i,$$
and the corresponding input nodes therefore have the same value function.

Let now $\mathring{v} \in \mathring{v}_1^{\mathrm{c}}$ be a computation node having predecessor node-vector
$$(\mathring{u}_1, \dots, \mathring{u}_k),$$
and suppose inductively that each $\mathring{u}_j$ and $\rho(\mathring{u}_j)$ have the same value function. Since ρ is a graph isomorphism, the predecessor node-vector of $\rho(\mathring{v})$ is
$$(\rho(\mathring{u}_1), \dots, \rho(\mathring{u}_k)).$$
Furthermore, computation-node labels are preserved, so
$$\lambda_2\big(\rho(\mathring{v})\big) = \lambda_1(\mathring{v}).$$
Hence, letting $\mathring{v}_\rho := \rho(\mathring{v})$ and $\mathring{v}_{i,\rho} := \rho(\mathring{u}_i)$, for every input $x \in \mathbb{R}^{d_1^{\mathrm{in}}}$, we have
$$\begin{aligned}\hat{v}_\rho(x) &= \hat{v}_\rho\big(\hat{u}_{i,\rho}(x), \dots, \hat{u}_{k,\rho}(x),\big) \\ &= \hat{v}_\rho\big(\hat{u}_i(x), \dots, \hat{u}_k(x)\big) \\ &= \hat{v}(x).\end{aligned}$$
This proves the claim for every computation node.

Finally, since ρ preserves the output node-vector, corresponding output coordinates of G_1 and G_2 have the same value function. Therefore $\hat{G}_1 = \hat{G}_2$, as required. □

Definition 148 (Renaming of computation nodes)**.** Let us consider a pair of computation graphs

$$G_1 = \langle V_1, E_1, \mathring{v}_1^{\mathrm{in}}, \mathring{v}_1^{\mathrm{c}}, \mathring{v}_1^{\mathrm{out}}, \lambda_1 \rangle, \qquad G_2 = \langle V_2, E_2, \mathring{v}_2^{\mathrm{in}}, \mathring{v}_2^{\mathrm{c}}, \mathring{v}_2^{\mathrm{out}}, \lambda_2 \rangle$$

whose input node-vectors coincide, namely $\mathring{v}_1^{\mathrm{in}} = \mathring{v}_2^{\mathrm{in}}$, and hence we write $\mathring{v}^{\mathrm{in}} := \mathring{v}_1^{\mathrm{in}} = \mathring{v}_2^{\mathrm{in}}$.

We say that G_1 and G_2 are *equal up to a renaming of computation nodes*, and write

$$G_1 \cong_{\mathrm{c}} G_2,$$

if there exists a computation-graph isomorphism $\rho : V_1 \to V_2$ that acts as the identity function on $\mathring{v}^{\mathrm{in}}$, namely

$$\rho\big((\mathring{v}^{\mathrm{in}})_i\big) = (\mathring{v}^{\mathrm{in}})_i \qquad \forall i \in [1..d^{\mathrm{in}}].$$

We also say that G_2 is G_1 *after renaming computation nodes*—and symmetrically that G_1 is G_2 *after renaming computation nodes*. *End of Definition 148.*

Definition 149 (Common-input union of computation graphs)**.** Let $G_1, \ldots, G_m$ be computation graphs given by

$$G_i = \langle V_i, E_i, \mathring{v}^{\mathrm{in}}, \mathring{v}_i^{\mathrm{c}}, \mathring{v}_i^{\mathrm{out}}, \lambda_i \rangle \qquad \forall i \in [1..m]$$

satisfying[3]

$$\mathrm{entries}(\mathring{v}_i^{\mathrm{c}}) \cap \mathrm{entries}\big(\mathring{v}_j^{\mathrm{c}}\big) = \emptyset \qquad \forall i, j \in [1..m] \text{ with } i \neq j.$$

Their *common-input union* is the computation graph

$$G_1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} G_m \;:=\; G = \langle V, E, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \rangle$$

where:

- the computation node-vector is
$$\mathring{v}^{\mathrm{c}} := \mathring{v}_1^{\mathrm{c}} \parallel \cdots \parallel \mathring{v}_m^{\mathrm{c}}$$
using $\parallel$ to denote concatenation of vectors;
- the multi-edge set is $E := \bigcup_{i=1}^{m} E_i$;
- the labelling function λ is defined for every computation node $\mathring{v} \in \mathring{v}^{\mathrm{c}}$ as:
$$\lambda(\mathring{v}) := \lambda_i(\mathring{v})$$
where $i \in [1..m]$ is the unique index such that $\mathring{v} \in \mathring{v}_i^{\mathrm{c}}$;
- the output node-vector is
$$\mathring{v}^{\mathrm{out}} := \mathring{v}_1^{\mathrm{out}} \parallel \cdots \parallel \mathring{v}_m^{\mathrm{out}}$$
again using $\parallel$ to denote concatenation of vectors.

End of Definition 149.

Definition 150 (Parametric computation graph)**.** Let $\mathcal{P}$ be a class of parametric Euclidean functions.

A *$\mathcal{P}$-parametric computation graph* is a pair

$$\mathcal{G} = \langle G, \mathring{v}^{\mathrm{par}} \rangle,$$

where G is a $\mathcal{P}$-computation graph, called the *underlying computation graph* of $\mathcal{G}$,

$$G = \langle V, E, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \rangle$$

and $\mathrm{entries}(\mathring{v}^{\mathrm{par}})$ is a node-vector of pair-wise distinct nodes from $\mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big)$; we call $\mathrm{entries}(\mathring{v}^{\mathrm{par}})$ the *parameter node-vector*, and we refer to its components as *parameter nodes*. The remaining input nodes form the *external-input-node set* $\mathring{v}^{\mathrm{ext}}$, obtained from $\mathring{v}^{\mathrm{in}}$ by removing the nodes occurring in $\mathring{v}^{\mathrm{par}}$; and we refer to the components of $\mathring{v}^{\mathrm{ext}}$ as *external-input nodes*. The parameter nodes and external-input nodes inherit their ordering from the input node-vector $\mathring{v}^{\mathrm{in}}$. We denote the corresponding node-vector presentations by $\mathring{v}^{\mathrm{par}}$ and $\mathring{v}^{\mathrm{ext}}$ respectively, and their dimensions are

$$d_{\mathrm{par}} := |\mathring{v}^{\mathrm{par}}|, \qquad d_{\mathrm{ext}} := |\mathring{v}^{\mathrm{ext}}|.$$

[3]The condition can always be guaranteed via a renaming of computation nodes.

The input dimension and output dimension of $\mathcal{G}$ is the input dimension d_{in} and output dimension d_{out} of G respectively. Thus, $d_{\text{in}} = d_{\text{ext}} + d_{\text{par}}$.

By identifying an input $x \in \mathbb{R}^{d_{\text{in}}}$ with a pair $(u, \theta) \in \mathbb{R}^{d_{\text{ext}}} \times \mathbb{R}^{d_{\text{par}}}$ according to the positions of the corresponding nodes in $\mathring{v}^{\text{in}}$, we write the value function of $\mathcal{G}$ as

$$\hat{\mathcal{G}} : \mathbb{R}^{d_{\text{ext}}} \times \mathbb{R}^{d_{\text{par}}} \to \mathbb{R}^{d_{\text{out}}}, \qquad \hat{\mathcal{G}}(u; \theta) := \hat{G}(u, \theta).$$

Similarly, for every node $\mathring{v} \in V$, we write $\hat{v}(u; \theta)$ for its value on input (u, θ).

We say that $\mathcal{G}$ *represents* the parametric function $\hat{\mathcal{G}}$. *End of Definition 150.*

Definition 151 (Common-input union of parametric computation graphs)**.** Let $m \in \mathbb{N}^{>0}$, and let us introduce parametric computation graphs

$$\mathcal{G}_i = \langle G_i, \mathring{v}^{\text{par}} \rangle \qquad \forall i \in [1..m],$$

having the same input node-vector and the same parameter-node set $\mathring{v}^{\text{par}}$, and assume that their underlying computation graphs $G_1, \ldots, G_m$ satisfy the conditions of Definition 149.

Their *common-input union* is the parametric computation graph

$$\mathcal{G}_1 \sqcup_{\text{in}} \cdots \sqcup_{\text{in}} \mathcal{G}_m := \big\langle G_1 \sqcup_{\text{in}} \cdots \sqcup_{\text{in}} G_m, \mathring{v}^{\text{par}} \big\rangle.$$

End of Definition 151.

Fact 41 (Correctness of common-input union)**.** *Let $Q_1, \ldots, Q_m$ be computation graphs satisfying the conditions of Definition 149, and suppose that Q_i represents*

$$q_i : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_i}, \qquad i \in [1..m].$$

Then, their common-input union

$$Q_1 \sqcup_{\text{in}} \cdots \sqcup_{\text{in}} Q_m$$

represents the function

$$q := (q_1, \ldots, q_m) : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_1 + \cdots + d_m}, \qquad q(x) := (q_1(x), \ldots, q_m(x)).$$

Similarly, let $\mathcal{Q}_1, \ldots, \mathcal{Q}_m$ be parametric computation graphs satisfying the conditions of Definition 151, and, for every $i \in [1..m]$, let

$$q_i : \mathbb{R}^{d_{\text{ext}}} \times \mathbb{R}^{d_{\text{par}}} \to \mathbb{R}^{d_i}$$

be the parametric function represented by $\mathcal{Q}_i$. Then, their common-input union

$$\mathcal{Q}_1 \sqcup_{\text{in}} \cdots \sqcup_{\text{in}} \mathcal{Q}_m$$

represents the parametric function

$$q := (q_1, \ldots, q_m) : \mathbb{R}^{d_{\text{ext}}} \times \mathbb{R}^{d_{\text{par}}} \to \mathbb{R}^{d_1 + \cdots + d_m}, \qquad q(u; \theta) := (q_1(u; \theta), \ldots, q_m(u; \theta)).$$

End of Fact 41.

Definition 152 (External-input restriction)**.** Let $\mathcal{G} = \langle G, \mathring{v}^{\text{par}} \rangle$ be a parametric computation graph having external-input node-vector $\mathring{v}^{\text{ext}}$ of dimension d_{ext}.

For every $u \in \mathbb{R}^{d_{\text{ext}}}$, the *external-input restriction of $\mathcal{G}$ at u* is the parametric computation graph

$$\mathcal{G}|_{\text{ext}=u} := \langle G|_{\mathring{v}^{\text{ext}}=u}, \mathring{v}^{\text{par}} \rangle.$$

Thus, all external-input nodes are to the have constant values given by the components of u, while the parameter nodes remain input nodes.

For every $u \in \mathbb{R}^{d_{\text{ext}}}$, and every node $\mathring{v}$ of G, we denote by $v|_{\text{ext}=u}$ the corresponding node of $\mathcal{G}|_{\text{ext}=u}$—namely, the same node $\mathring{v}$ but now seen as a node of the graph $\mathcal{G}|_{\text{ext}=u}$.
End of Definition 152.

Fact 42 (Correctness of external-input restriction)**.** *Assume the context of Definition 152,*

For every $u \in \mathbb{R}^{d_{\text{ext}}}$, the following holds:

- *the restriction $\mathcal{G}|_{\mathrm{ext}=u}$ has no external-input nodes and has parameter-node vector $\mathring{v}^{\mathrm{par}}$;*
- *for every node $\mathring{v}$ of G, the following two identities hold:*

$$\hat{v}|_{\mathrm{ext}=u}(\theta) = \hat{v}(u;\theta) \quad \textit{and} \quad \hat{\mathcal{G}}|_{\mathrm{ext}=u}(\theta) = \hat{\mathcal{G}}(u;\theta) \qquad \forall \theta \in \mathbb{R}^{d_{\mathrm{par}}}.$$

End of Fact 42.

Definition 153 (Parameter independence)**.** Let $\mathcal{G} = \langle G, \mathring{v}^{\mathrm{par}} \rangle$ be a parametric computation graph.

A node $\mathring{v}$ of G is *parameter-independent* if it is independent of $\mathring{v}^{\mathrm{par}}$ in the sense of Definition 144; equivalently, we have that $\mathring{v}$ is not a parameter node and is not reachable by a directed path from any parameter node.

A node-vector or node-matrix presentation of G is *parameter-independent* if each of its scalar nodes is parameter-independent.

The parametric computation graph $\mathcal{G}$ is *parameter-independent* if its output node-vector is parameter-independent. *End of Definition 153.*

Definition 154 (Canonical constant computation graph)**.** Let $c \in \mathbb{R}^n$ and let $\mathring{v}^{\mathrm{in}}$ be a vector of d_{in} pair-wise distinct input nodes.

The *canonical constant computation graph representing c over $\mathring{v}^{\mathrm{in}}$* is the computation graph

$$G_{\mathrm{c}}[c, \mathring{v}^{\mathrm{in}}] := \langle V, E, \mathring{v}^{\mathrm{in}}, \mathring{c}, \mathring{c}, \lambda \rangle$$

having components as follows:

- the node-vector $\mathring{c}$ is a vector of n pair-wise distinct nodes not occurring in $\mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big)$;
- the node set is $V := \mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big) \cup \mathrm{entries}(\mathring{c})$;
- the edge set is $E := \big\{ () \to \mathring{c}_i \;\big|\; i \in [1..n] \big\}$—it consists of multi-edges of zero input-arity;
- the labelling function is $\lambda : \mathrm{entries}(\mathring{c}) \to \mathrm{entries}(c)$ given by $\lambda(\mathring{c}_i) := (c)_i$ for every $i \in [1..n]$.

Note that the labelling function λ assigns nodes to constant functions, which have zero input-arity, hence correctly matching the input-arity of the introduced multi-edges. *End of Definition 154.*

Fact 43. *In the context of Definition 154,*

- *both the computation-node vector and the output node-vector of $G_{\mathrm{c}}[c, \mathring{v}^{\mathrm{in}}]$ coincide with $\mathring{c}$;*
- *the value function of the graph is the following constant function:*

$$\hat{G}_{\mathrm{c}}[c, \mathring{v}^{\mathrm{in}}](x) = c \qquad \forall x \in \mathbb{R}^{d_{\mathrm{in}}}.$$

End of Fact 43.

Fact 44 (Canonical constant graphs for dynamics)**.** *Assume the context of Definition 154.*

Let D be parametric dynamics having input space $U \subseteq \Omega_U = \mathbb{R}^{d_{\mathrm{in}}}$. Let $\mathring{v}^{\mathrm{in}}$ be a node-vector of d_{in} pair-wise distinct input nodes.

Then, the graph $G_{\mathrm{c}}[c, \mathring{v}^{\mathrm{in}}]$ is a canonical constant computation graph representing the constant c over the input ambient-space Ω_U of D. End of Fact 44.

Definition 155 (Parametric canonical constant computation graph)**.** Let $c \in \mathbb{R}^n$. We define two variants of canonical constant parametric computation graphs.

- Let $\mathcal{H} = \langle H, \mathring{v}^{\mathrm{par}} \rangle$ be a parametric computation graph, and let $\mathring{v}^{\mathrm{in}}$ be its input node-vector. The *canonical constant parametric computation graph representing c over $\mathcal{H}$* is

$$G^{\mathrm{p}}_{\mathrm{c}}[c, \mathcal{H}] := \big\langle G_{\mathrm{c}}[c, \mathring{v}^{\mathrm{in}}],\ \mathring{v}^{\mathrm{par}} \big\rangle.$$

- Let $d_{\mathrm{in}}, d_{\mathrm{par}} \in \mathbb{N}$, let $\mathring{v}^{\mathrm{in}}$ be a node-vector of d_{in} pair-wise distinct input nodes, and let $\mathring{v}^{\mathrm{par}}$ be a node-vector of d_{par} pair-wise distinct nodes that are not in $\mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big)$. The *canonical constant parametric computation graph representing* c *over the input node-vector* $\mathring{v}^{\mathrm{in}}$ *and parameter node-vector* $\mathring{v}^{\mathrm{par}}$ is
$$G_{\mathrm{c}}^{\mathrm{p}}[c, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{par}}] := \big\langle G_{\mathrm{c}}[c, \mathring{v}^{\mathrm{par}}],\ \mathring{v}^{\mathrm{par}} \big\rangle.$$

End of Definition 155.

Fact 45. *Assume the context of Definition 155. We have the following facts for canonical constant graphs.*

The canonical graph $G_{\mathrm{c}}^{\mathrm{p}}[c, \mathcal{H}]$ *satisfies the following:*

- *it has the same external-input nodes and parameter nodes as* $\mathcal{H}$*;*
- *it is parameter-independent;*
- *its value function satisfies the following identity:*
$$\hat{G}_{\mathrm{c}}^{\mathrm{p}}[c, \mathcal{H}](u; \theta) = c \qquad \forall u \in \mathbb{R}^{d_{\mathrm{ext}}},\ \forall \theta \in \mathbb{R}^{d_{\mathrm{par}}}.$$

The canonical constant graph $G_{\mathrm{c}}^{\mathrm{p}}[c, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{par}}]$ *satisfies the following:*

- *it has no external-input nodes;*
- *it has parameter-node vector* $\mathring{v}^{\mathrm{par}}$*;*
- *it is parameter-independent;*
- *its value function satisfies the following identity:*
$$\hat{G}_{\mathrm{c}}^{\mathrm{p}}[c, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{par}}](u; \theta) = c \qquad \forall \theta \in \Theta. \forall u \in \mathbb{R}^{d_{\mathrm{ext}}},\ \forall \theta \in \Theta.$$

End of Fact 45.

Fact 46 (Parametric canonical constant graphs for parametric dynamics)**.** *Assume the context of Definition 155.*

Let D *be parametric dynamics having input space* $U \subseteq \Omega_U = \mathbb{R}^{d_{\mathrm{in}}}$ *and parameter space* $\Theta \subseteq \Omega_\Theta = \mathbb{R}^{d_{\mathrm{par}}}$*. Let* $\mathring{v}^{\mathrm{in}}$ *be a node-vector of* d_{in} *pair-wise distinct input nodes, and let* $\mathring{v}^{\mathrm{par}}$ *be a node-vector of* d_{par} *pair-wise distinct nodes that are not in* $\mathrm{entries}\big(\mathring{v}^{\mathrm{in}}\big)$*.*

Then, the graph $G_{\mathrm{c}}^{\mathrm{p}}[c, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{par}}]$ *is a canonical constant parametric computation graph representing the constant* c *over the joint input-parameter ambient-space* $\Omega_U \times \Omega_\Theta$ *of* D*.* End of Fact 46.

Definition 156 (Computation-graph family)**.** For X a set and $\mathcal{P}$ a class of Euclidean functions, a *$\mathcal{P}$-computation-graph family with index set X* is map from X to non-empty sets of $\mathcal{P}$-computation graphs, and we write it as
$$\mathcal{G} = \big(\mathcal{G}[x]\,\big)_{x \in X},$$
meaning that $\mathcal{G}[x]$ is the set of computation graphs for $x \in X$.

For $\mathcal{Q}$ and $\mathcal{P}$ classes of Euclidean functions, a *$\mathcal{P}$-computation-graph family for $\mathcal{Q}$* is a $\mathcal{P}$-computation-graph family with index set $\mathcal{Q}$ such that, for every function $q \in \mathcal{Q}$, every computation graph $G \in \mathcal{G}[q]$ satisfies the following conditions jointly:

(1) the input dimension of G equals the input arity of q;

(2) the output dimension of G equals the output arity of q;

(3) G represents q.

In this case, for every $q \in \mathcal{Q}$, we call $\mathcal{G}[q]$ the *computation-graph family for q induced by $\mathcal{G}$*, or simply the *computation-graph family for q* when $\mathcal{G}$ is clear from context.

For $\mathcal{Q}^{\mathrm{p}}$ and $\mathcal{P}^{\mathrm{p}}$ classes of parametric Euclidean functions, a *$\mathcal{P}^{\mathrm{p}}$-parametric computation-graph family for $\mathcal{Q}^{\mathrm{p}}$* is a $\mathcal{P}^{\mathrm{p}}$-computation-graph family with index set $\mathcal{Q}^{\mathrm{p}}$ such that, for every function $q \in \mathcal{Q}^{\mathrm{p}}$, every computation graph $G \in \mathcal{G}[q]$ is a parametric computation graph satisfying the following conditions jointly:

(1) the input dimension of G equals the input arity of q;

(2) the parameter dimension of G equals the parameter arity of q;

(3) the output dimension of G equals the output arity of q;

(4) G represents q.

In this case, for every $q \in \mathcal{Q}^{\mathrm{p}}$, we call $\mathcal{G}[q]$ the *parametric computation-graph family for q induced by $\mathcal{G}$*, or simply the *parametric computation-graph family for q* when $\mathcal{G}$ is clear from context. *End of Definition 156.*

Definition 157 (Graph-family admissibility)**.** Let $\mathcal{Q}$ be a class of Euclidean functions, and let $\mathcal{G}$ be a computation-graph family for $\mathcal{Q}$. A pseudo-derivative operator $\tilde{\mathsf{D}}$ is *admissible for $\mathcal{G}$* if, for every function $q \in \mathcal{Q}$ and every graph $G \in \mathcal{G}[q]$, we have that $\tilde{\mathsf{D}}$ is admissible for G. The definition applies seamlessly to the parametric case. *End of Definition 157.*

Definition 158 (Graph selectors)**.** Let $\mathcal{Q}$ be a class of Euclidean functions, and let $\mathcal{G}$ be a computation-graph family for $\mathcal{Q}$. A *graph selector for $\mathcal{Q}$ over $\mathcal{G}$* is a map Γ satisfying

$$\Gamma(q) \in \mathcal{G}[q] \qquad \forall q \in \mathcal{Q}.$$

The definition applies seamlessly to the parametric case. *End of Definition 158.*

E.2 Pseudo-derivative Operators

Definition 159 (Pseudo-derivative operators)**.** Let $q : X \to Y$ be a Euclidean function having input arity d_{in} and output arity d_{out}. A *pseudo-derivative operator* is an operator $\tilde{\mathsf{D}}$ that maps Euclidean functions to Euclidean matrix-valued functions. The operator $\tilde{\mathsf{D}}$ is *applicable* to q when $\tilde{\mathsf{D}}$ is defined for q and it yiels a Euclidean function of the following form:

$$\tilde{\mathsf{D}}q : X \to \mathbb{R}^{\mathrm{ar}_{\mathrm{out}}(q) \times \mathrm{ar}_{\mathrm{in}}(q)}.$$

In this case, the function $\tilde{\mathsf{D}}q$ is the *pseudo-derivative function of q under* $\tilde{\mathsf{D}}$; we may also call $\tilde{\mathsf{D}}q$ the *pseudo-Jacobian* of q under $\tilde{\mathsf{D}}$, and the *pseudo-gradient* of q under $\tilde{\mathsf{D}}$ when $d_{\mathrm{out}} = 1$.

Let $\mathcal{Q}$ be a class of Euclidean functions. A *pseudo-derivative operator* applicable to $\mathcal{Q}$ is a pseudo-derivative operator that is applicable to every function $q \in \mathcal{Q}$. *End of Definition 159.*

Definition 160 (Positional partial pseudo-derivatives)**.** Let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator, and let

$$q : \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_k} \to \mathbb{R}^{d_{\mathrm{out}}}$$

be a function to which $\tilde{\mathsf{D}}$ is applicable.

For

$$x = (x_1, \ldots, x_k), \qquad x_i \in \mathbb{R}^{d_i},$$

write the pseudo-derivative matrix according to the input-block decomposition as

$$\tilde{\mathsf{D}}q(x) = \begin{pmatrix} J_1(x) & \cdots & J_k(x) \end{pmatrix}, \qquad J_i(x) \in \mathbb{R}^{d_{\mathrm{out}} \times d_i}.$$

For every $i \in [1..k]$, the *pseudo-derivative of q with respect to its i-th input block* is the function

$$\tilde{\mathsf{D}}_i q$$

defined by

$$\tilde{\mathsf{D}}_i q(x) := J_i(x).$$

Hence,

$$\tilde{\mathsf{D}}q(x) = \begin{pmatrix} \tilde{\mathsf{D}}_1 q(x) & \cdots & \tilde{\mathsf{D}}_k q(x) \end{pmatrix}.$$

End of Definition 160.

Definition 161 (Named partial pseudo-derivatives)**.** Let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator, and let

$$q : \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_k} \to \mathbb{R}^{d_{\text{out}}}$$

be a function to which $\tilde{\mathsf{D}}$ is applicable.

When evaluating the pseudo-derivative at a point

$$x = (x_1, \ldots, x_k),$$

we use a *named partial pseudo-derivative notation* in which the subscript identifies one of the input blocks of the evaluation point.

More precisely, for every $i \in [1..k]$, we write

$$\tilde{\mathsf{D}}_{x_i} q\big(x_1, \ldots, x_k\big) := \tilde{\mathsf{D}}_i q\big(x_1, \ldots, x_k\big).$$

Thus,

$$\tilde{\mathsf{D}}_{x_i} q\big(x_1, \ldots, x_k\big) \in \mathbb{R}^{d_{\text{out}} \times d_i}$$

denotes the block of the pseudo-derivative matrix at $(x_1, \ldots, x_k)$ corresponding to variations of the input block x_i.

Hence, $\tilde{\mathsf{D}}_i q$ denotes a block pseudo-derivative function indexed by the position of the corresponding input block, whereas

$$\tilde{\mathsf{D}}_{x_i} q\big(x_1, \ldots, x_k\big)$$

is a pointwise notation in which the differentiated input block is identified by its name at the evaluation point.

When an input block at the evaluation point is itself given by the value of a function or expression, we use the same convention.

More precisely, suppose that

$$q : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_{\text{out}}},$$

and consider an evaluation point of the form

$$(g(x), y).$$

The fully explicit named partial pseudo-derivative notation for the first input block is

$$\tilde{\mathsf{D}}_{g(x)} q\big(g(x), y\big).$$

When there is no ambiguity, we abbreviate the name of this input block by the symbol of the function producing it, and write

$$\tilde{\mathsf{D}}_g q\big(g(x), y\big) := \tilde{\mathsf{D}}_{g(x)} q\big(g(x), y\big).$$

More generally, named subscripts may abbreviate the expression defining the corresponding input block whenever the association between the name and the block at the evaluation point is clear from context. *End of Definition 161.*

Definition 162 (Graph-admissible pseudo-derivative operator)**.** A pseudo-derivative operator $\tilde{\mathsf{D}}$ is *admissible for a computation graph* G if it is applicable to every label of a computation node of G. A pseudo-derivative operator $\tilde{\mathsf{D}}$ is *admissible for a parametric computation graph* $\mathcal{G} = \langle G, \mathring{v}^{\text{par}} \rangle$ if it is admissible for its underlying computation graph G. *End of Definition 162.*

Definition 163 (Graph-induced pseudo-derivatives)**.** Let G be a computation graph having input dimension d_{in}, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator that is admissible for G.

The *pseudo-derivative operator induced by* G *with local pseudo-derivative operator* $\tilde{\mathsf{D}}$ is the operator $\tilde{\mathsf{D}}^G$ defined first on the nodes of G and then on G itself, as follows:

- for every input node $\mathring{v}_i^{\text{in}}$, we define its pseudo-derivative as:

$$\tilde{\mathsf{D}}^G \mathring{v}_i^{\text{in}}(x) \;:=\; D\hat{v}_i^{\text{in}}(x) \;=\; e_i^\top \;\in \mathbb{R}^{1 \times d_{\text{in}}} \qquad\qquad \forall x \in \mathbb{R}^{\text{in}};$$

  where e_i is the i-th standard basis vector of $\mathbb{R}^{d_{\text{in}}}$,

- for every zero-arity computation node $\mathring{v}$, we define its pseudo-derivative as:

$$\tilde{\mathsf{D}}^G \mathring{v}(x) \;:=\; 0 \;\in \mathbb{R}^{1 \times d_{\text{in}}} \qquad\qquad \forall x \in \mathbb{R}^{\text{in}};$$

- for every node-vector $\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k)$, we define its pseudo-derivative by vertically stacking the pseudo-derivatives of its components:

$$\tilde{\mathsf{D}}^G \mathring{u}(x) \;:=\; \begin{pmatrix} \tilde{\mathsf{D}}^G \mathring{u}_1(x) \\ \vdots \\ \tilde{\mathsf{D}}^G \mathring{u}_k(x) \end{pmatrix} \qquad\qquad \forall x \in \mathbb{R}^{\text{in}},$$

  where the right-hand side is understood as a block-column matrix.

- for every positive-arity computation node $\mathring{v}$ having input node-vector $\mathring{u}$, we define its pseudo-derivative as:

$$\tilde{\mathsf{D}}^G \mathring{v}(x) \;:=\; \tilde{\mathsf{D}} \check{v}\big(\hat{u}(x)\big) \cdot \tilde{\mathsf{D}}^G \mathring{u}(x) \qquad\qquad \forall x \in \mathbb{R}^{\text{in}};$$

- finally, for $\mathring{v}^{\text{out}}$ the output node-vector of G, we define the pseudo-derivative of G as:

$$\tilde{\mathsf{D}} G := \tilde{\mathsf{D}}^G \mathring{v}^{\text{out}}.$$

The definition applies to every parametric computation graph $\mathcal{G} = (G, \mathring{v}^{\text{par}})$ by applying it to its underlying computation graph G. In particular,

$$\tilde{\mathsf{D}} \mathcal{G} := \tilde{\mathsf{D}} G.$$

End of Definition 163.

Proposition 23 (Correctness of graph-induced ordinary derivatives)**.** *Let* D *be the ordinary derivative operator, let* $\mathcal{P}_{\text{diff}}$ *be a class of differentiable Euclidean functions, let* G *be an* $\mathcal{P}_{\text{diff}}$*-computation graph, and let* $f := \hat{G}$ *be the function represented by* G*. Then,*

- *the operator* D *is a pseudo-derivative operator that is admissible for* G*;*
- *the function* f *is differentiable and the graph-induced pseudo-derivative of* G *coincides with the ordinary derivative of* f*:*

$$\mathsf{D} G = \mathsf{D} f.$$

Proof. Since G is an $\mathcal{P}_{\text{diff}}$-computation graph and every function in $\mathcal{P}_{\text{diff}}$ is differentiable, the ordinary derivative operator D is admissible for G.

We prove, following a topological ordering of the nodes of G, that every node $\mathring{v}$ has a differentiable value function $\hat{v}$ and satisfies

$$\tilde{\mathsf{D}}^G \mathring{v}(x) = \mathsf{D} \hat{v}(x) \qquad \forall x \in \mathbb{R}^{d_{\text{in}}}, \tag{25}$$

where the local pseudo-derivative operator $\tilde{\mathsf{D}}$ is taken to be D.

First, let $\mathring{v}_i^{\text{in}}$ be an input node. Its value function is the coordinate projection

$$\hat{v}_i^{\text{in}}(x) = (x)_i,$$

which is differentiable, and by Definition 163,

$$\tilde{\mathsf{D}}^G \mathring{v}_i^{\text{in}}(x) = e_i^\top = \mathsf{D} \hat{v}_i^{\text{in}}(x).$$

Now let $\mathring{v}$ be a zero-arity computation node. Its value function is constant as a function of the input x, and is therefore differentiable with

$$\mathsf{D} \hat{v}(x) = 0.$$

Again by Definition 163,

$$\tilde{\mathsf{D}}^G \mathring{v}(x) = 0 = \mathsf{D} \hat{v}(x).$$

Finally, let $\mathring{v}$ be a positive-arity computation node with predecessor node-vector

$$\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k).$$

By the induction hypothesis, every value function $\hat{u}_i$ is differentiable and

$$\tilde{\mathsf{D}}^G \mathring{u}_i(x) = \mathsf{D}\hat{u}_i(x).$$

Hence the node-vector value function

$$\hat{u} = (\hat{u}_1, \dots, \hat{u}_k)$$

is differentiable, with derivative obtained by vertically stacking the derivatives of its components:

$$\mathsf{D}\hat{u}(x) = \begin{pmatrix} \mathsf{D}\hat{u}_1(x) \\ \vdots \\ \mathsf{D}\hat{u}_k(x) \end{pmatrix}.$$

By the induction hypothesis and the definition of the pseudo-derivative of a node-vector,

$$\tilde{\mathsf{D}}^G \mathring{u}(x) = \mathsf{D}\hat{u}(x).$$

Since G is an $\mathcal{P}_{\text{diff}}$-computation graph, its node label $\check{v}$ belongs to $\mathcal{P}_{\text{diff}}$ and is therefore differentiable. Furthermore,

$$\hat{v} = \check{v} \circ \hat{u}.$$

Thus $\hat{v}$ is differentiable and, by the chain rule,

$$\begin{aligned} \mathsf{D}\hat{v}(x) &= \mathsf{D}\check{v}\big(\hat{u}(x)\big) \cdot \mathsf{D}\hat{u}(x) \\ &= \mathsf{D}\check{v}\big(\hat{u}(x)\big) \cdot \tilde{\mathsf{D}}^G \mathring{u}(x) \\ &= \tilde{\mathsf{D}}^G \mathring{v}(x), \end{aligned}$$

where the last equality follows from Definition 163. This proves (25) for every node of G.

Applying the same componentwise argument to the output node-vector $\mathring{v}^{\text{out}}$, we obtain that

$$f = \hat{G} = \hat{v}^{\text{out}}$$

is differentiable and

$$\begin{aligned} \mathsf{D}f(x) &= \mathsf{D}\hat{v}_{\text{out}}(x) \\ &= \tilde{\mathsf{D}}^G \mathring{v}^{\text{out}}(x) \\ &= \mathsf{D}G(x). \end{aligned}$$

Therefore

$$\mathsf{D}G = \mathsf{D}f,$$

as required. □

Lemma 6 (Parameter-independent graphs have zero parameter derivative)**.** *Let $\mathcal{G}$ be a parametric computation graph, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator that is admissible for $\mathcal{G}$.*

Let u and θ be an external input and parameter values for $\mathcal{G}$, respectively.

- *If $\mathring{v}$ is a parameter-independent node-vector or node-matrix presentation of $\mathcal{G}$, then*

$$\tilde{\mathsf{D}}\mathring{v}|_{\text{ext}=u}(\theta) = 0.$$

- *If $\mathcal{G}$ is parameter-independent, then*

$$\tilde{\mathsf{D}}\mathcal{G}|_{\text{ext}=u}(\theta) = 0.$$

Proof. We first prove that every parameter-independent node $\mathring{v}$ of $\mathcal{G}$ satisfies

$$\tilde{\mathsf{D}}v|_{\text{ext}=u}(\theta) = 0. \tag{26}$$

We proceed according to a topological ordering of the nodes of $\mathcal{G}|_{\text{ext}=u}$.

Suppose first that $\mathring{v}$ is an external-input node of $\mathcal{G}$. By Definition 152, $v|_{\mathrm{ext}=u}$ is a zero-arity computation node. Hence, by Definition 163,

$$\tilde{\mathsf{D}} v|_{\mathrm{ext}=u}(\theta) = 0.$$

Now let $\mathring{v}$ be a parameter-independent computation node of $\mathcal{G}$, with input node-vector

$$\mathring{w} = (\mathring{w}_1, \ldots, \mathring{w}_k).$$

Every predecessor $\mathring{w}_i$ is parameter-independent. Indeed, if $\mathring{w}_i$ were a parameter node or reachable from a parameter node, then $\mathring{v}$ would also be reachable from a parameter node, contradicting its parameter-independence.

Therefore, by the induction hypothesis,

$$\tilde{\mathsf{D}} w_i|_{\mathrm{ext}=u}(\theta) = 0 \qquad \forall i \in [1..k].$$

Thus, by the definition of the pseudo-derivative of a node-vector,

$$\tilde{\mathsf{D}} w|_{\mathrm{ext}=u}(\theta) = 0.$$

Since external-input restriction leaves $\mathring{v}$ and its label unchanged, the recursive definition of the graph-induced pseudo-derivative gives

$$\tilde{\mathsf{D}} v|_{\mathrm{ext}=u}(\theta) \;=\; \tilde{\mathsf{D}}\check{v}\big(\hat{w}_u(\theta)\big) \cdot \tilde{\mathsf{D}} w|_{\mathrm{ext}=u}(\theta) \;=\; 0 \qquad \text{where} \quad w_u := w|_{\mathrm{ext}=u}.$$

This proves (26) for every parameter-independent node.

If $\mathring{v}$ is a parameter-independent node-vector presentation, then every scalar node occurring in $\mathring{v}$ is parameter-independent. Hence (26) holds coordinatewise, and therefore

$$\tilde{\mathsf{D}} \mathring{v}|_{\mathrm{ext}=u}(\theta) = 0.$$

The same argument applies entrywise to a parameter-independent node-matrix presentation. This proves the first claim.

Finally, if $\mathcal{G}$ is parameter-independent, then its output node-vector $\mathring{v}^{\mathrm{out}}$ is parameter-independent. By the first claim,

$$\tilde{\mathsf{D}} \mathring{v}^{\mathrm{out}}|_{\mathrm{ext}=u}(\theta) = 0.$$

Since $\mathring{v}^{\mathrm{out}}|_{\mathrm{ext}=u}$ is the output node-vector of $\mathcal{G}|_{\mathrm{ext}=u}$, it follows that

$$\tilde{\mathsf{D}} \mathcal{G}|_{\mathrm{ext}=u}(\theta) = 0.$$

This concludes the proof. □

E.3 Selected Pseudo-Derivative Operators

Definition 164 (Selected pseudo-derivative operator)**.** Let $\mathcal{P}$ be a class of Euclidean functions, let $\mathcal{F}$ be the closure of $\mathcal{F}$, let $\mathcal{G}$ be a $\mathcal{P}$-computation graph family for $\mathcal{F}$, let Γ be a graph selector over $\mathcal{G}$, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator that is applicable to $\mathcal{P}$.

The *graph-selected pseudo-derivative operator induced by* $(\tilde{\mathsf{D}}, \Gamma)$, is defined by

$$\tilde{\mathsf{D}}^{\Gamma} q := \tilde{\mathsf{D}}^{\Gamma(q)} q \qquad \forall q \in \mathcal{Q}.$$

The following terminology applies:

- $\tilde{\mathsf{D}}$ is the *local pseudo-derivative operator* of $\tilde{\mathsf{D}}^{\Gamma}$;
- Γ is the *graph selector* of $\tilde{\mathsf{D}}^{\Gamma}$, or simply its *selector*;
- $\Gamma(q)$ is the *selected computation graph of* q *under* $\tilde{\mathsf{D}}^{\Gamma}$;
- we abbreviate the term *graph-selected pseudo-derivative operator* as *selected pseudo-derivative operator* and also as *SPD-operator*.

End of Definition 164.

Definition 165 (Named partial selected pseudo-derivatives)**.** Let $\tilde{\mathsf{D}}^{\Gamma}$ be a selected pseudo-derivative operator, and let

$$q : \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_k} \to \mathbb{R}^{d_{\text{out}}}$$

be a function to which $\tilde{\mathsf{D}}^{\Gamma}$ is applicable.

We extend the notation of Definition 161 to selected pseudo-derivatives as follows.

For every $i \in [1..k]$,

$$\tilde{\mathsf{D}}^{\Gamma}_{i} q$$

denotes the block selected pseudo-derivative function corresponding to the ith input block, and

$$\tilde{\mathsf{D}}^{\Gamma}_{x_i} q\big(x_1, \ldots, x_k\big) := \tilde{\mathsf{D}}^{\Gamma}_{i} q\big(x_1, \ldots, x_k\big)$$

denotes its value at the point $(x_1, \ldots, x_k)$, with the differentiated input block identified by its name.

The same abbreviation convention applies when an input block is given by the value of a function or expression. In particular, for an evaluation point

$$(g(x), y),$$

we write

$$\tilde{\mathsf{D}}^{\Gamma}_{g} q\big(g(x), y\big) := \tilde{\mathsf{D}}^{\Gamma}_{g(x)} q\big(g(x), y\big).$$

More generally, named subscripts may abbreviate the expression defining the corresponding input block whenever the association between the name and the block at the evaluation point is clear from context. *End of Definition 165.*

E.4 Intensional-Derivative Operators

Definition 166 (Intensional-derivative operators)**.** Let q be a Euclidean PAP function having input arity d_{in} and output arity d_{out}. An *intensional-derivative operator* applicable to q is a pseudo-derivative operator $\tilde{\mathsf{D}}$ such that $\tilde{\mathsf{D}}q$ is an intensional derivative of q, namely $\tilde{\mathsf{D}}q \in \partial_{\bullet} q$. In this case, the function $\tilde{\mathsf{D}}q$ is the *intensional-derivative function of q under* $\tilde{\mathsf{D}}$; we may also call $\tilde{\mathsf{D}}q$ the *intensional-Jacobian* of q under $\tilde{\mathsf{D}}$, and the *intensional-gradient* of q under $\tilde{\mathsf{D}}$ when $d_{\text{out}} = 1$.

Let $\mathcal{Q}$ be a class of Euclidean PAP functions. An *intensional-derivative operator* applicable to $\mathcal{Q}$ is a pseudo-derivative operator that is an intensional-derivative operator for every function $q \in \mathcal{Q}$.
End of Definition 166.

E.5 Graph-Induced Intensional-Derivative Operators

Definition 167 (Intensional graph-admissibility)**.** A pseudo-derivative operator $\tilde{\mathsf{D}}$ is *intensionally-admissible for a computation graph G* if it is an intensional-derivative operator for every computation-node label $\check{v}$ of G.

A pseudo-derivative operator $\tilde{\mathsf{D}}$ is *intensionally-admissible for a parametric computation graph* $\mathcal{G} = (G, \mathring{v}^{\text{par}})$ if it is intensionally-admissible for G. *End of Definition 167.*

Proposition 24 (Correctness of graph-induced pseudo-derivatives)**.** *Let $\mathcal{P}_{\text{PAP}}$ be a class of PAP Euclidean functions, let G be a $\mathcal{P}_{\text{PAP}}$-computation graph, and let*

$$f := \hat{G}$$

be the function represented by G.

If $\tilde{\mathsf{D}}$ is intensionally-admissible for G, then f is a PAP function and the graph-induced pseudo-derivative induced by G is an intensional derivative of f, namely, $\tilde{\mathsf{D}}G \in \partial_{\bullet} f$.

Proof. Since $\tilde{\mathsf{D}}$ is intensionally-admissible for G, it is an intensional-derivative operator for every computation-node label of G. In particular, $\tilde{\mathsf{D}}$ is applicable to every computation-node label, and hence it is admissible for G. Therefore, the graph-induced pseudo-derivative $\tilde{\mathsf{D}}^G$ is well-defined.

We prove, following a topological ordering of the nodes of G, that every node $\mathring{v}$ satisfies the following two properties:

(1) its value function $\hat{v}$ is PAP;

(2) its graph-induced pseudo-derivative is an intensional derivative of its value function:

$$\tilde{\mathsf{D}}^G \mathring{v} \in \partial_\bullet \hat{v}. \tag{27}$$

First, let $\mathring{v}_i^{\mathrm{in}}$ be an input node. Its value function is the coordinate projection

$$\hat{v}_i^{\mathrm{in}}(x) = (x)_i.$$

This function is analytic and therefore PAP. Furthermore, it admits a one-piece PAP representation whose analytic representative is the coordinate projection itself. The intensional derivative induced by this representation is its ordinary derivative

$$x \longmapsto e_i^\top.$$

By Definition 163,

$$\tilde{\mathsf{D}}^G \mathring{v}_i^{\mathrm{in}}(x) = e_i^\top.$$

Hence

$$\tilde{\mathsf{D}}^G \mathring{v}_i^{\mathrm{in}} \in \partial_\bullet \hat{v}_i^{\mathrm{in}}.$$

Now let $\mathring{v}$ be a zero-arity computation node. Its value function is constant as a function of the input of G. Hence it is analytic and therefore PAP. A one-piece PAP representation of this constant function has identically-zero derivative, so

$$x \longmapsto 0$$

is an intensional derivative of $\hat{v}$. By Definition 163,

$$\tilde{\mathsf{D}}^G \mathring{v}(x) = 0.$$

Thus

$$\tilde{\mathsf{D}}^G \mathring{v} \in \partial_\bullet \hat{v}.$$

Finally, let $\mathring{v}$ be a positive-arity computation node having predecessor node-vector

$$\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_k).$$

By the induction hypothesis, for every $i \in [1..k]$, the value function $\hat{u}_i$ is PAP and

$$\tilde{\mathsf{D}}^G \mathring{u}_i \in \partial_\bullet \hat{u}_i.$$

By Lemma 1, the node-vector value function

$$\hat{u} = (\hat{u}_1, \ldots, \hat{u}_k)$$

is PAP, and the vertically stacked function

$$x \longmapsto \begin{pmatrix} \tilde{\mathsf{D}}^G \mathring{u}_1(x) \\ \vdots \\ \tilde{\mathsf{D}}^G \mathring{u}_k(x) \end{pmatrix}$$

is an intensional derivative of $\hat{u}$. By Definition 163, this function is exactly $\tilde{\mathsf{D}}^G \mathring{u}$. Therefore,

$$\tilde{\mathsf{D}}^G \mathring{u} \in \partial_\bullet \hat{u}. \tag{28}$$

Since G is a $\mathcal{P}_{\mathrm{PAP}}$-computation graph, its computation-node label satisfies

$$\check{v} \in \mathcal{P}_{\mathrm{PAP}}.$$

Hence $\check{v}$ is PAP.

Furthermore, since $\tilde{\mathsf{D}}$ is intensionally-admissible for G, it is an intensional-derivative operator for $\check{v}$. Thus

$$\tilde{\mathsf{D}}\check{v} \in \partial_\bullet \check{v}. \tag{29}$$

By the definition of the value function of a computation node,

$$\hat{v} = \check{v} \circ \hat{u}.$$

Since both $\hat{u}$ and $\check{v}$ are PAP, Proposition 9 of Lee et al. [43] implies that $\hat{v}$ is PAP.

Furthermore, by (28), (29), and the chain rule for intensional derivatives given by Proposition 10 of Lee et al. [43], the function

$$x \longmapsto \tilde{\mathsf{D}}\check{v}\big(\hat{u}(x)\big) \cdot \tilde{\mathsf{D}}^G\mathring{u}(x)$$

is an intensional derivative of

$$\check{v} \circ \hat{u} = \hat{v}.$$

By Definition 163, this function is precisely $\tilde{\mathsf{D}}^G\mathring{v}$. Hence

$$\tilde{\mathsf{D}}^G\mathring{v} \in \partial_\bullet \hat{v}.$$

This establishes both induction claims for every node of G.

It remains to consider the output node-vector

$$\mathring{v}^{\mathrm{out}} = (\mathring{v}_1^{\mathrm{out}}, \ldots, \mathring{v}_{d_{\mathrm{out}}}^{\mathrm{out}}).$$

By the result just proved, for every $i \in [1..d_{\mathrm{out}}]$, the value function $\hat{v}_i^{\mathrm{out}}$ is PAP and

$$\tilde{\mathsf{D}}^G\mathring{v}_i^{\mathrm{out}} \in \partial_\bullet \hat{v}_i^{\mathrm{out}}.$$

Applying Lemma 1 once more shows that

$$f = \hat{G} = {}^{\circ}\hat{\mathrm{out}}\mathring{v}$$

is PAP and that

$$\tilde{\mathsf{D}}^G\mathring{v}^{\mathrm{out}} \in \partial_\bullet f.$$

Finally, by Definition 163,

$$\tilde{\mathsf{D}}G = \tilde{\mathsf{D}}^G\mathring{v}^{\mathrm{out}},$$

and therefore

$$\tilde{\mathsf{D}}G \in \partial_\bullet f.$$

Hence f is PAP and the graph-induced pseudo-derivative induced by G is an intensional derivative of f, as required. □

Corollary 11 (Correctness from an intensional-derivative operator)**.** *Let $\mathcal{P}_{\mathrm{PAP}}$ be a class of PAP Euclidean functions, let G be an $\mathcal{P}_{\mathrm{PAP}}$-computation graph, and let*

$$f := \hat{G}$$

be the function represented by G.

If $\tilde{\mathsf{D}}$ is an intensional-derivative operator for $\mathcal{P}_{\mathrm{PAP}}$, then f is a PAP function and the graph-induced pseudo-derivative induced by G is an intensional derivative of f, namely $\tilde{\mathsf{D}}G \in \partial_\bullet f$.

Proof. Since G is an $\mathcal{P}_{\mathrm{PAP}}$-computation graph, every computation-node label belongs to $\mathcal{P}_{\mathrm{PAP}}$. Since $\tilde{\mathsf{D}}$ is an intensional-derivative operator for $\mathcal{P}_{\mathrm{PAP}}$, it is an intensional-derivative operator for every computation-node label of G. Hence $\tilde{\mathsf{D}}$ is intensionally-admissible for G by Definition 167.

The result follows from Proposition 24. □

Corollary 12 (Correctness for parametric computation graphs)**.** *Let $\mathcal{P}_{\mathrm{PAP}}$ be a class of PAP Euclidean functions, let $\mathcal{G} = (G, \mathring{v}^{\mathrm{par}})$ be a parametric computation graph whose underlying computation graph G is a $\mathcal{P}_{\mathrm{PAP}}$-computation graph, and let $f := \hat{\mathcal{G}}$ be the parametric function represented by $\mathcal{G}$.*

If $\tilde{\mathsf{D}}$ is intensionally-admissible for $\mathcal{G}$, then f is a PAP function and the graph-induced pseudo-derivative induced by $\mathcal{G}$ is an intensional derivative of f, namely $\tilde{\mathsf{D}}\mathcal{G} \in \partial_\bullet f$.

Proof. Since $\tilde{\mathsf{D}}$ is intensionally-admissible for $\mathcal{G}$, Definition 167 implies that $\tilde{\mathsf{D}}$ is intensionally-admissible for its underlying computation graph G.

Since G is a $\mathcal{P}_{\mathrm{PAP}}$-computation graph, Proposition 24 therefore implies that $\hat{G}$ is PAP and

$$\tilde{\mathsf{D}}G \in \partial_{\bullet}\hat{G}.$$

By the definition of a parametric computation graph, $\hat{\mathcal{G}}$ is the value function $\hat{G}$ with its input coordinates distinguished into external-input and parameter coordinates. Hence

$$f = \hat{\mathcal{G}} = \hat{G},$$

under this identification.

Furthermore, by Definition 163,

$$\tilde{\mathsf{D}}\mathcal{G} = \tilde{\mathsf{D}}G.$$

Therefore f is PAP and

$$\tilde{\mathsf{D}}\mathcal{G} \in \partial_{\bullet} f,$$

as required. □

E.6 Selected Intensional-Derivative Operators

Definition 168 (Selected intensional-derivative operator)**.** Let $\mathcal{Q}$ be a class of PAP Euclidean functions, and let $\tilde{\mathsf{D}}^{\Gamma}$ be a selected pseudo-derivative operator applicable to $\mathcal{Q}$.

The operator $\tilde{\mathsf{D}}^{\Gamma}$ is a *selected intensional-derivative operator*for $\mathcal{Q}$ if it is an intensional-derivative operator for $\mathcal{Q}$. *End of Definition 168.*

Proposition 25 (Intensional-admissibility implies SID)**.** *Let $\mathcal{P}_{\mathrm{PAP}}$ be a class of PAP Euclidean functions, let $\mathcal{Q}$ be a class of Euclidean functions, and let $\tilde{\mathsf{D}}^{\Gamma}$ be an SPD-operator applicable to $\mathcal{Q}$ such that*

$$\Gamma(q)$$

is a $\mathcal{P}_{\mathrm{PAP}}$-computation graph for every $q \in \mathcal{Q}$.

If $\tilde{\mathsf{D}}$ is intensionally-admissible for $\mathrm{Im}(\Gamma)$, then every function in $\mathcal{Q}$ is PAP and $\tilde{\mathsf{D}}^{\Gamma}$ is an SID-operator for $\mathcal{Q}$.

Proof. Let $q \in \mathcal{Q}$ be arbitrary, and let

$$G_q := \Gamma(q)\,.$$

By hypothesis, G_q is a $\mathcal{P}_{\mathrm{PAP}}$-computation graph. Moreover, since

$$G_q \in \mathrm{Im}(\Gamma),$$

the hypothesis of intensional admissibility implies that $\tilde{\mathsf{D}}$ is intensionally-admissible for G_q.

Since G_q represents q, Proposition 24 implies that q is PAP and that

$$\tilde{\mathsf{D}}G_q \in \partial_{\bullet} q.$$

By Definition 164,

$$\tilde{\mathsf{D}}^{\Gamma} q = \tilde{\mathsf{D}}G_q = \tilde{\mathsf{D}}\,\Gamma(q)\,.$$

Therefore

$$\tilde{\mathsf{D}}^{\Gamma} q \in \partial_{\bullet} q.$$

Since $q \in \mathcal{Q}$ was arbitrary, every function in $\mathcal{Q}$ is PAP and $\tilde{\mathsf{D}}^{\Gamma}$ is an intensional-derivative operator for $\mathcal{Q}$. Hence, by Definition 168, it is an SID-operator for $\mathcal{Q}$. □

Remark 18 (Dependence on the selected computation graph). Fix a local pseudo-derivative operator $\tilde{\mathsf{D}}$. Different computation graphs representing the same function may induce different intensional derivatives of that function.

An SID-operator $\tilde{\mathsf{D}}^{\Gamma}$ includes a graph selector Γ that fixes one computation graph for each function in its domain. Consequently, $\tilde{\mathsf{D}}^{\Gamma}$ uniquely determines a selected intensional derivative for every function to which it is applicable.

At the same time, two SID-operators $\tilde{\mathsf{D}}^{\Gamma_1}$ and $\tilde{\mathsf{D}}^{\Gamma_2}$ may have the same local pseudo-derivative operator $\tilde{\mathsf{D}}$ but different graph selectors. They may therefore produce different intensional derivatives of the same function; namely, it is possible that

$$\tilde{\mathsf{D}}^{\Gamma_1} f \neq \tilde{\mathsf{D}}^{\Gamma_2} f$$

for some function f. *End of Remark 18.*

Example 3 (Different selected intensional derivatives for min and max functions)**.** We exemplify Remark 18.

Consider the pointwise maximum functions $\max_n : \mathbb{R}^n \to \mathbb{R}$ from Definition 132,

$$\max_n(x_1, \ldots, x_n) := \bigoplus_{i=1}^{n} x_i,$$

and let $\tilde{\mathsf{D}}$ be the MinMax pseudo-derivative operator from Definition 228. In particular,

$$\tilde{\mathsf{D}} \max_2(a, a) = \left(\frac{1}{2}, \frac{1}{2}\right) \qquad \forall a \in \mathbb{R}.$$

The function $\max_3$ admits, among others, computation graphs G_1 and G_2 corresponding respectively to

$$\max_3(x_1, x_2, x_3) = \max_2\big(\max_2(x_1, x_2), x_3\big),$$
$$\max_3(x_1, x_2, x_3) = \max_2\big(x_1, \max_2(x_2, x_3)\big).$$

Both are computation graphs whose computation-node labels are instances of the PAP function $\max_2$.

Let Γ_1 and Γ_2 be graph selectors satisfying

$$\Gamma_1[\max_3] = G_1, \qquad \Gamma_2[\max_3] = G_2.$$

Since $\tilde{\mathsf{D}}$ is an intensional-derivative operator for the local primitive $\max_2$ (as shown in Appendix M), it is intensionally-admissible for both G_1 and G_2. Hence the corresponding selected pseudo-derivative operators are SID-operators by Proposition 25.

Consider now a point

$$x = (a, a, a), \qquad a \in \mathbb{R}.$$

At this point, every maximum node in both graphs is evaluated at a tie.

- For G_1, the inner maximum assigns weight $1/2$ to each of x_1 and x_2, while the outer maximum assigns weight $1/2$ to the inner maximum and $1/2$ to x_3. Consequently,

$$\tilde{\mathsf{D}}^{\Gamma_1} \max_3(a, a, a) = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right).$$

- For G_2, the inner maximum assigns weight $1/2$ to each of x_2 and x_3, while the outer maximum assigns weight $1/2$ to x_1 and $1/2$ to the inner maximum. Consequently,

$$\tilde{\mathsf{D}}^{\Gamma_2} \max_3(a, a, a) = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right).$$

Therefore,

$$\tilde{\mathsf{D}}^{\Gamma_1} \max_3(a, a, a) \neq \tilde{\mathsf{D}}^{\Gamma_2} \max_3(a, a, a).$$

Thus, two SID-operators having the same local pseudo-derivative operator $\tilde{\mathsf{D}}$ but different graph selectors may select different intensional derivatives of the same function, even though their selected computation graphs represent the same function $\max_3$. *End of Example 3.*

E.7 Analytic Extensions of Computation Graphs

We introduce a construction for extending computation graphs by an analytic outer function. It will be used to represent analytic parameter-dependent maps, output maps, and losses, and to propagate graph-induced pseudo-derivatives through them.

Definition 169 (Analytic extension of computation graphs)**.** Let $Q_1, \dots, Q_k$, for $k \in \mathbb{N}^{>0}$, be computation graphs satisfying the conditions of Definition 149, with common input dimension d_{in}, respective output dimensions $d_1, \dots, d_k$, and represented functions

$$q_i : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_i}, \qquad i \in [1..k].$$

Let

$$g = (g_1, \dots, g_{d_{\mathrm{out}}}) : \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_k} \to \mathbb{R}^{d_{\mathrm{out}}}$$

be analytic.

Let

$$Q^{\sqcup} := Q_1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} Q_k, \qquad \mathring{u} := \mathring{v}^{\mathrm{out},1} \parallel \cdots \parallel \mathring{v}^{\mathrm{out},k},$$

where $\mathring{v}^{\mathrm{out},i}$ is the output node-vector inherited from Q_i.

The *analytic extension of* $(Q_1, \dots, Q_k)$ *by* g, denoted by

$$G := (Q_1, \dots, Q_k) \triangleright g,$$

is obtained from $Q^{\sqcup}$ by adjoining fresh computation nodes

$$\mathring{w} := (\mathring{w}_1, \dots, \mathring{w}_{d_{\mathrm{out}}}),$$

with predecessor node-vector $\mathring{u}$ and labels

$$\breve{w}_j := g_j, \qquad j \in [1..d_{\mathrm{out}}],$$

and by taking $\mathring{w}$ as the output node-vector. All nodes, multi-edges, labels, and the input node-vector inherited from $Q^{\sqcup}$ are otherwise unchanged. *End of Definition 169.*

Proposition 26 (Correctness of analytic graph extensions)**.** *In the context of Definition 169, the graph*

$$G = (Q_1, \dots, Q_k) \triangleright g$$

represents

$$F : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}}, \qquad F(x) := g\big(q_1(x), \dots, q_k(x)\big).$$

Proof. For every $x \in \mathbb{R}^{d_{\mathrm{in}}}$, the predecessor node-vector of the newly introduced output nodes satisfies

$$\hat{u}(x) = (q_1(x), \dots, q_k(x)),$$

because the corresponding components are the output node-vectors of $Q_1, \dots, Q_k$. Since $\breve{w}_j = g_j$ for every $j \in [1..d_{\mathrm{out}}]$, we obtain

$$\hat{G}(x) = g\big(q_1(x), \dots, q_k(x)\big) = F(x).$$

□

Fact 47 (Preservation of computation-graph primitive classes)**.** *In the context of Definition 169, let $\mathcal{P}$ be a class of Euclidean functions. If every Q_i is a $\mathcal{P}$-computation graph and*

$$g_j \in \mathcal{P} \qquad \forall j \in [1..d_{\mathrm{out}}],$$

then

$$(Q_1, \dots, Q_k) \triangleright g$$

is a $\mathcal{P}$-computation graph. End of Fact 47.

Proof. The extension retains all computation-node labels of $Q_1, \dots, Q_k$ and introduces only the labels $g_1, \dots, g_{d_{\mathrm{out}}}$, which belong to $\mathcal{P}$ by hypothesis. □

Proposition 27 (Analytic chain rule for graph extensions)**.** *In the context of Definition 169, let*

$$G := (Q_1, \dots, Q_k) \triangleright g,$$

and let $\tilde{\mathsf{D}}$ *be a pseudo-derivative operator admissible for* G*. If*

$$\tilde{\mathsf{D}} g_j = \mathsf{D} g_j \qquad \forall j \in [1..d_{\mathrm{out}}],$$

then, for every $x \in \mathbb{R}^{d_{\mathrm{in}}}$*,*

$$\begin{aligned} \tilde{\mathsf{D}} G(x) &= \mathsf{D} g\big(q_1(x), \dots, q_k(x)\big) \begin{pmatrix} \tilde{\mathsf{D}} Q_1(x) \\ \vdots \\ \tilde{\mathsf{D}} Q_k(x) \end{pmatrix} \\ &= \sum_{i=1}^{k} \mathsf{D}_{q_i} g\big(q_1(x), \dots, q_k(x)\big) \tilde{\mathsf{D}} Q_i(x). \end{aligned} \tag{30}$$

Proof. Fix $x \in \mathbb{R}^{d_{\mathrm{in}}}$. Since the extension does not modify any node inherited from $Q_1, \dots, Q_k$, Definition 163 gives

$$\tilde{\mathsf{D}}^{G} \mathring{u}(x) = \begin{pmatrix} \tilde{\mathsf{D}} Q_1(x) \\ \vdots \\ \tilde{\mathsf{D}} Q_k(x) \end{pmatrix}.$$

Moreover,

$$\hat{u}(x) = (q_1(x), \dots, q_k(x)).$$

Since the new output nodes have labels $g_1, \dots, g_{d_{\mathrm{out}}}$ and $\tilde{\mathsf{D}} g_j = \mathsf{D} g_j$, Definition 163 yields

$$\tilde{\mathsf{D}} G(x) = \mathsf{D} g\big(q_1(x), \dots, q_k(x)\big) \begin{pmatrix} \tilde{\mathsf{D}} Q_1(x) \\ \vdots \\ \tilde{\mathsf{D}} Q_k(x) \end{pmatrix}.$$

Expanding $\mathsf{D} g$ by input blocks gives the second equality in (30). □

F Properties: Expressivity, BIBS, BIBO, BIBSD, BIBLG, and nnVSD

Table of Contents

F.1 Expressivity 112
F.2 BIBS and BIBO 114
F.3 BIBSD: Bounded-Input Bounded-State-Derivative 115
F.4 BIBLG: Bounded-Input Bounded-Loss-Gradient 117
F.5 nnVSD: Non-necessarily Vanishing State Derivative 119

This appendix introduces the formal properties used throughout the paper to characterise the expressive power, stability, and gradient behaviour of recurrent systems and their parametric variants. The properties are organised around three increasingly derivative-sensitive levels. At the level of values, BIBS and BIBO formalise boundedness of state and output trajectories under bounded inputs (Definitions 176 and 177). At the level of parameter sensitivity, BIBSD requires derivatives of states with respect to parameters to remain bounded, while BIBLG formulates the corresponding property directly for gradients of training objectives. Finally, nnVSD captures a complementary phenomenon: state-to-state derivatives need not necessarily vanish with the sequence length (Definition 197).

Several variants are required because the results in the paper quantify over different objects. For parametric systems, we distinguish weak properties, where bounds may depend on the particular bounded input trajectory and initial state, from uniform properties, where a single bound applies on every bounded input, state, and parameter set; see Definitions 178 and 179 for BIBS and BIBO, and Definitions 180 and 182 for BIBSD. For derivatives that may fail to exist everywhere, we also consider the almost-everywhere variants, including Definitions 181 and 183. In addition, selected intensional derivatives require explicit control of the computation graph used to represent a function. This leads to selected and graph-uniform formulations, in particular Definitions 184, 186, 193, and 195. These distinctions allow the statements in the main text to separate properties of the represented recurrent system from properties of a particular differentiable or piecewise-differentiable computational realisation.

The appendix is organised as follows. We first define formal-language expressivity and its η-finite variant in Definitions 173 and 174. We then introduce BIBS and BIBO and their weak and uniform parametric variants in Definitions 176–179. Next, we develop BIBSD, progressing from weak and almost-everywhere formulations to uniform, selected, and graph-uniform variants; see in particular Definitions 180, 181, 182, 183, and 186. We subsequently define samples, losses, and objective functions in Definition 189, and use them to formulate BIBLG, ae-BIBLG, selected BIBLG, and graph-uniform selected BIBLG in Definitions 191, 192, 193, and 195. The connection between the uniform state-derivative properties and loss-gradient bounds is summarised in Remark 23. Finally, Definition 197 introduces nnVSD, which formalises the existence of bounded-input trajectories along which state-to-state derivatives remain bounded away from zero uniformly in time.

F.1 Expressivity

Definition 84 gives the underlying notion of exact implementation and recognition. We refine it by requiring the decoder to separate the outputs produced by the system from differently decoded outputs by a fixed positive metric margin.

For this section, let us fix an arbitrary system

$$S = \langle X, U, f, x_0, Y, h \rangle .$$

Definition 170 (Decoder margin)**.** Let $\gamma \in \mathbb{R}^{\geq 0}$ and let $\hat{Y} \subseteq Y$. We say that a decoder $\mathrm{dec} : Y \to \Gamma$[4] *admits a margin* γ *on* $\hat{Y}$ if, for every pair of $\hat{y} \in \hat{Y}$ and $y \in Y$,[5]

$$d_Y(\hat{y}, y) < \gamma \quad \Longrightarrow \quad \mathrm{dec}(\hat{y}) = \mathrm{dec}(y) .$$

End of Definition 170.

[4] The notion of *decoder* is introduced by Definition 83. Recall that Γ is a finite alphabet.
[5] Recall that d_Y denotes the metric of the output space Y.

Definition 171 (Implementation with a margin)**.** Given a function $F : \Sigma^+ \to \Gamma$, we define the following notions of implementation of F by system S with a margin.

SubDefinition 171.1. Given an encoder $\mathrm{enc} : \Sigma \to U$ and a decoder $\mathrm{dec} : Y \to \Gamma$,[6] system S *implements F with margin $\gamma \in \mathbb{R}^{>0}$, encoder* enc*, and decoder* dec, if (i) S implements F with encoder enc and decoder dec, and (ii) dec admits a γ-margin on the output set Y_S given by

$$Y_S := \left\{ S(\mathrm{enc}(w)) \,:\, w \in \Sigma^+ \right\}.$$

SubDefinition 171.2. System S *can-implement F with margin $\gamma \in \mathbb{R}^{>0}$* if there exist an encoder enc and decoder dec such that S implements F with margin γ, encoder enc, and decoder dec.

SubDefinition 171.3. System S *can-implement F with positive margin* if S can-implement F with some positive margin $\gamma \in \mathbb{R}^{>0}$.

End of Definition 171.

Definition 172 (Recognition with a margin)**.** Given a language $L \subseteq \Sigma^*$, we define the following notions of recognition of L by system S with a margin.

SubDefinition 172.1. Given an encoder $\mathrm{enc} : \Sigma \to U$ and a decoder $\mathrm{dec} : Y \to \{0, 1\}$,system S *recognises language L with margin $\gamma \in \mathbb{R}^{>0}$, encoder* enc*, and decoder* dec if S implements the characteristic function $\mathbf{1}_L$ with margin γ, encoder enc, and decoder dec.

SubDefinition 172.2. System S *recognises language L with margin $\gamma \in \mathbb{R}^{>0}$* if there exist an encoder $\mathrm{enc} : \Sigma \to U$ and a decoder $\mathrm{dec} : Y \to \{0, 1\}$,such that S recognises language L with margin γ, encoder enc, and decoder dec.

SubDefinition 172.3. System S *can-recognise language L with positive margin* if S can-recognise L with some positive margin $\gamma \in \mathbb{R}^{>0}$.

End of Definition 172.

F.1.1 Expressivity of a Class of Systems

Definition 173 (Ordinary expressivity)**.** We define the notion of *formal-language expressivity* for classes of systems, parameteric systems, and classes of parametric systems:

- the *formal-language expressivity* of a class $\mathcal{S}$ of systems is the set consisting of each language L for which that exists a system $S \in \mathcal{S}$ that can-recognise L;
- the *formal-language expressivity* of a parameteric system S is the set consisting of each language L for which there exists an instance $S[\theta]$ of S that can-recognise L;
- the *formal-language expressivity* of a class $\mathcal{S}$ of parameteric systems is the set consisting of each language L for which there exists a parametric system $S \in \mathcal{S}$ admitting an instance $S[\theta]$ that can-recognise L.

We often say *expressivity* instead of *formal-language expressivity*. *End of Definition 173.*

Definition 174 (η-expressivity)**.** We define the notion of *formal-language η-expressivity* for classes of systems, parameteric systems, and classes of parametric systems:

- the *formal-language η-expressivity* of a class $\mathcal{S}$ of systems is the set consisting of each language L for which there exists an η-finite system $S \in \mathcal{S}$ that can-recognise L;
- the *formal-language η-expressivity* of a parameteric system S is the set consisting of each language L for which there exists an η-finite instance $S[\theta]$ of S that can-recognise L;

[6] The notions of *encoder* and *decoder* are introduced by Definition 83. Recall that Σ and Γ are finite alphabets called the *input alphabet* and *ouput alphabet*, respectively.

- the *formal-language η-expressivity* of a class $\mathcal{S}$ of parameteric systems is the set consisting of each language L for which there exists a parameteric system $S \in \mathcal{S}$ admitting an instance $S[\theta]$ that can-recognise L.

We often say *η-expressivity* instead of *formal-language η-expressivity*. *End of Definition 174.*

Definition 175 (Positive-margin expressivity)**.** We define the notion of *positive-margin formal-language expressivity* for classes of systems, parameteric systems, and classes of parametric systems:

- the *positive-margin formal-language expressivity* of a class $\mathcal{S}$ of systems is the set consisting of each language L for which there exists a system $S \in \mathcal{S}$ that can-recognise L with positive margin;
- the *positive-margin formal-language expressivity* of a parameteric system S is the set consisting of each language L for wichh there exists an instance $S[\theta]$ of S that can-recognise L with positive margin;
- the *positive-margin formal-language expressivity* of a class $\mathcal{S}$ of parameteric systems is the set consisting of each language L for which there exists a parametric system $S \in \mathcal{S}$ admitting an instance $S[\theta]$ that can-recognise L with positive margin.

We often say *positive-margin expressivity* instead of *positive-margin formal-language expressivity*. *End of Definition 175.*

F.2 BIBS and BIBO

Definition 176. Euclidean dynamics $D = \langle X, U, f \rangle$ satisfy *bounded-input bounded-state (BIBS)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for every choice of an initial state $\dot{x} \in X_{\leq r}$ and of an input sequence $(u_t)_{t \in \mathbb{T}} \in U^{\omega}_{\leq r}$, the corresponding state sequence

$$(x_t)_{t \in \mathbb{T}} := \big(\check{D}(\dot{x}, u_{[1\,..\,t]})\big)_{t \in \mathbb{T}} \qquad \text{satisfies} \qquad \sup_{t \in \mathbb{T}} \|x_t\| \leq B\,.$$

A Euclidean system S satisfies *BIBS* if its dynamics $\mathrm{dyn}(S)$ satisfy BIBS. *End of Definition 176.*

Definition 177. A Euclidean system $S = \langle X, U, f, x^{\mathrm{init}}, Y, h \rangle$ satisfies *bounded-input bounded-output (BIBO)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for every choice of an initial state $\dot{x} \in X_{\leq r}$ and of an input sequence $(u_t)_{t \in \mathbb{T}} \in U^{\omega}_{\leq r}$, the corresponding output sequence

$$\begin{aligned} &(y_t)_{t \in \mathbb{T}} := \big(h(x_{t-1}, u_t, x_t)\big)_{t \in \mathbb{T}} \\ \text{with} \quad &(x_t)_{t \in \mathbb{T}} := \big(\check{\mathrm{dyn}}(\mathrm{S})(\dot{x}, u_{[1\,..\,t]})\big)_{t \in \mathbb{T}} \end{aligned} \qquad \text{satisfies} \qquad \sup_{t \in \mathbb{T}} \|y_t\| \leq B_r\,.$$

End of Definition 177.

Definition 178 (Weak parametric and class-wise BIBS and BIBO)**.** The non-parametric BIBS and BIBO properties are extended to parametric objects in a weak, instance-wise sense as follows:

- Euclidean parametric dynamics D satisfy *weak bounded-input bounded-state (wBIBS)* if

$$D[\theta] \text{ satisfies BIBS} \qquad \forall \theta \in \Theta;$$

- a Euclidean parametric system S satisfies *weak bounded-input bounded-state (wBIBS)* if

$$S[\theta] \text{ satisfies BIBS} \qquad \forall \theta \in \Theta;$$

- a Euclidean parametric system S satisfies *weak bounded-input bounded-output (wBIBO)* if

$$S[\theta] \text{ satisfies BIBO} \qquad \forall \theta \in \Theta;$$

- a class of Euclidean dynamics satisfies BIBS if every member of the class satisfies the corresponding property;
- a class of Euclidean dynamics or systems satisfies BIBS or BIBO if every member of the class satisfies the corresponding property;

- a class of Euclidean parametric dynamics satisfies wBIBS if every member of the class satisfies the corresponding property.
- a class of Euclidean parametric systems satisfies wBIBS or wBIBO if every member of the class satisfies the corresponding property.

End of Definition 178.

Remark 19. The qualifier *weak* refers to the absence of uniformity across parameter instances and bounded input trajectories. In particular, the BIBS or BIBO bound of an instance may depend on the parameter value, the initial state, and the bounded input sequence.

Definition 179 (Uniform parametric and class-wise BIBS and BIBO)**.** We define the following BIBS and BIBO notions.

SubDefinition 179.1. Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ satisfy *uniform bounded-input bounded-state (uBIBS)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for every choice of parameter values $\theta \in \Theta_{\leq r}$, of an initial state $\dot{x} \in X_{\leq r}$, and of an input sequence $u \in U^{\omega}_{\leq r}$, the corresponding state sequence

$$(x_t)_{t\in\mathbb{T}} := \left(\check{D}[\theta](\dot{x}, u_{[1\,..\,t]})\right)_{t\in\mathbb{T}} \qquad \text{satisfies} \qquad \sup_{t\in\mathbb{T}} \|x_t\| \leq B_r\,.$$

SubDefinition 179.2. A Euclidean parametric system S satisfies *uBIBS* if its dynamics $\mathrm{dyn}(S)$ satisfy uBIBS.

SubDefinition 179.3. A Euclidean parametric system $S = \langle X, U, f, x^{\mathrm{init}}, Y, h; \Theta^f, \Theta^h \rangle$ satisfies *uniform bounded-input bounded-output (uBIBO)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for every choice of dynamics-parameter values $\theta_f \in \Theta^f_{\leq r}$, of output-parameter values $\theta_h \in \Theta^h_{\leq r}$, of an initial state $\dot{x} \in X_{\leq r}$, and of an input sequence $u \in U^{\omega}_{\leq r}$, the corresponding output sequence

$$\begin{aligned} &(y_t)_{t\in\mathbb{T}} := h(x_{t-1}, u_t, x_t; \theta_h) \\ \text{with } &(x_t)_{t\in\mathbb{T}} := \left(\mathrm{dyn}(\check{\mathrm{S}})[\theta_{\mathrm{f}}](\dot{x}, u_{[1\,..\,t]})\right)_{t\in\mathbb{T}} \end{aligned} \qquad \text{satisfies} \qquad \sup_{t\in\mathbb{T}} \|y_t\| \leq B_r\,.$$

SubDefinition 179.4. A class of Euclidean parametric dynamics or parametric systems satisfies uBIBS or uBIBO if every member of the class satisfies the corresponding property.

End of Definition 179.

Remark 20 (Relationship between weak and uniform BIBS/BIBO). Uniform boundedness implies weak boundedness:

$$\begin{aligned} \mathrm{uBIBS} &\Longrightarrow \mathrm{wBIBS}, \\ \mathrm{uBIBO} &\Longrightarrow \mathrm{wBIBO}. \end{aligned}$$

The converses need not hold, since under wBIBS and wBIBO the bound may vary with the parameter instance and the bounded input trajectory.

F.3 BIBSD: Bounded-Input Bounded-State-Derivative

Definition 180 (wBIBSD)**.** Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ satisfy *weak bounded-input bounded-state-derivative (wBIBSD)* if, for every constant $r \in \mathbb{R}^{\geq 0}$ and for every choice of an initial state $\dot{x} \in X_{\leq r}$ and of an input sequence $u \in U^{\omega}_{\leq r}$, there exists a constant $B_{r,u,\dot{x}} \in \mathbb{R}^{\geq 0}$ such that, for every $t \in \mathbb{T}$,

$$\check{D}[\dot{x}, u, t] \text{ is differentiable on } \Theta_{\leq r} \qquad \text{and} \qquad \sup_{\theta\in\Theta_{\leq r}} \left\|\mathsf{D}\check{D}[\dot{x}, u, t](\theta)\right\| \leq B_{r,u,\dot{x}}\,.$$

End of Definition 180.

Definition 181 (ae-wBIBSD)**.** Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ satisfy *almost-everywhere weak bounded-input bounded-state-derivative (ae-wBIBSD)* if, for every constant $r \in \mathbb{R}^{\geq 0}$, every choice of an initial state $\dot{x} \in X_{\leq r}$, and of an input sequence $u \in U^{\omega}_{\leq r}$, there exist

$$\textit{a constant } B_r^{\dot{x},u} \in \mathbb{R}^{\geq 0}\,, \qquad \textit{a full-measure subset } \hat{\Theta}_r^{\dot{x},u} \subseteq \Theta_{\leq r}\,,$$

such that, for every $t \in \mathbb{T}$,

$$\check{D}[\dot{x}, u, t] \textit{ is differentiable on } \hat{\Theta}_r^{\dot{x},u} \qquad \text{and} \qquad \sup_{\theta \in \hat{\Theta}_r^{\dot{x},u}} \left\| \mathsf{D}\check{D}[\dot{x}, u, t](\theta) \right\| \leq B_r^{\dot{x},u}\,.$$

End of Definition 181.

Remark 21. The adjective *weak* indicates that the derivative bound may depend on the bounded input trajectory u and the initial state $\dot{x}$. The bound is nevertheless uniform over the bounded parameter set $\Theta_{\leq r}$ and over time.

Definition 182 (uBIBSD)**.** Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ satisfy *uniform bounded-input bounded-state-derivative (uBIBSD)* if, for every constant $r \in \mathbb{R}^{\geq 0}$, there exists a constant $B_r \in \mathbb{R}^{\geq 0}$ such that, for every choice of an initial state $\dot{x} \in X_{\leq r}$ and of an input sequence $u \in U^{\omega}_{\leq r}$, we have that, for every $t \in \mathbb{T}$,

$$\check{D}[\dot{x}, u, t] \textit{ is differentiable on } \Theta_{\leq r}\theta \qquad \text{and} \qquad \sup_{\theta \in \Theta_{\leq r}} \left\| \mathsf{D}\check{D}[\dot{x}, u, t](\theta) \right\| \leq B_r\,.$$

End of Definition 182.

Definition 183 (ae-uBIBSD)**.** Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ satisfy *almost-everywhere uniform bounded-input bounded-state-derivative (ae-uBIBSD)* if, for every constant $r \in \mathbb{R}^{\geq 0}$, there exists a constant $B_r \in \mathbb{R}^{\geq 0}$ such that, for every choice of an initial state $\dot{x} \in X_{\leq r}$ and of an input sequence $u \in U^{\omega}_{\leq r}$, there exists

$$\textit{a full-measure subset } \hat{\Theta}_{r,u,\dot{x}} \subseteq \Theta_{\leq r}$$

such that, for every $t \in \mathbb{T}$,

$$\check{D}[\dot{x}, u, t] \textit{ is differentiable on } \hat{\Theta}_{r,u,\dot{x}} \qquad \text{and} \qquad \sup_{\theta \in \hat{\Theta}_{r,u,\dot{x}}} \left\| \mathsf{D}\check{D}[\dot{x}, u, t](\theta) \right\| \leq B_r\,.$$

End of Definition 183.

The notions of selected intensional derivative operator, selected computation graph, computation-graph family, and graph selector are introduced in Section E.4.

Definition 184 (Selected uniform BIBSD)**.** Let $\tilde{\mathsf{D}}$ be a selected intensional derivative operator. Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ satisfy *$\tilde{\mathsf{D}}$-uniform bounded-input bounded-state-derivative ($\tilde{\mathsf{D}}$-uBIBSD)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for choice of an initial state every $\dot{x} \in X_{\leq r}$ and an input sequence $u \in U^{\omega}_{\leq r}$, we have that, for every $t \in \mathbb{T}$,

$$\tilde{\mathsf{D}} \textit{ is applicable to } \check{D}[\dot{x}, u, t]|_{\Theta_{\leq r}} \qquad \text{and} \qquad \sup_{\theta \in \Theta_{\leq r}} \left\| \tilde{\mathsf{D}}\check{D}[\dot{x}, u, t](\theta) \right\| \leq B_r\,.$$

End of Definition 184.

Definition 185 (State computation-graph family)**.** For D Euclidean parametric dynamics, a *state computation-graph family* for D is a computation-graph family $\mathcal{G}$ for the family of parameter-to-state functions $\check{D}$. *End of Definition 185.*

Definition 186 (Graph-uniform selected uBIBSD)**.** Let D be Euclidean parametric dynamics, let $\mathcal{G}$ be a state computation-graph family for D, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator that is intensionally-admissible for $\mathcal{G}$.

We say that D satisfies *$\mathcal{G}$-uniform selected bounded-input bounded-state-derivative ($\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \left\| \tilde{\mathsf{D}}G(\theta) \right\| \leq B_r \qquad \forall u \in U^{\omega}_{\leq r},\ \forall \dot{x} \in X_{\leq r},\ \forall t \in \mathbb{T},\ \forall G \in \mathcal{G}\Big[\check{D}[\dot{x}, u, t]\Big]\,.$$

End of Definition 186.

Remark 22 (Relationship between weak and uniform BIBSD). The uniform properties imply their weak counterparts:

$$\text{uBIBSD} \implies \text{wBIBSD},$$
$$\text{ae-uBIBSD} \implies \text{ae-wBIBSD}.$$

The converses need not hold because the constants in wBIBSD and ae-wBIBSD may depend on the bounded input trajectory and the initial state.

Definition 187 (System-wise BIBSD properties)**.** The preceding state-derivative properties are extended to an arbitrary Euclidean parametric system S as follows:

- system S satisfies wBIBSD or uBIBSD if its dynamics $\mathrm{dyn}(S)$ satisfy the corresponding property and its output function h is differentiable jointly in its non-parameter and parameter arguments;
- system S satisfies ae-wBIBSD or ae-uBIBSD if its dynamics $\mathrm{dyn}(S)$ satisfy the corresponding property and its output function h is almost-everywhere differentiable jointly in its non-parameter and parameter arguments;
- given $\tilde{\mathsf{D}}$ a selected intensional derivative operator, we say that system S satisfies $\tilde{\mathsf{D}}$-uBIBSD if its dynamics $\mathrm{dyn}(S)$ satisfy $\tilde{\mathsf{D}}$-uBIBSD;
- given a state computation-graph family $\mathcal{G}$ for the dynamics $\mathrm{dyn}(S)$, and a pseudo-derivative operator $\tilde{\mathsf{D}}$ that is intensionally-admissible for $\mathcal{G}$, we say that system S satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD if its dynamics $\mathrm{dyn}(S)$ satisfy $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD.

End of Definition 187.

Definition 188 (Class-wise BIBSD properties)**.** A class of Euclidean parametric dynamics or parametric systems satisfies wBIBSD, ae-wBIBSD, uBIBSD, ae-uBIBSD, $\tilde{\mathsf{D}}$-uBIBSD, or $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD if every member of the class satisfies the corresponding property. *End of Definition 188.*

Remark 23 (Uniform properties for loss-gradient bounds). The uniform properties are the variants requiring a single bound on every bounded input and parameter set. This requirement is in line with the loss-gradient notions introduced below in §F.4. In particular, uBIBS holding jointly with uBIBSD, ae-uBIBSD, $\tilde{\mathsf{D}}$-uBIBSD, or $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD, provides the state and state-parameter derivative uniformity required by the corresponding BIBLG, ae-BIBLG, $\tilde{\mathsf{D}}$-BIBLG, or $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG lifting results.

F.4 BIBLG: Bounded-Input Bounded-Loss-Gradient

Definition 189 (Samples, loss and objective functions)**.** Let S be a Euclidean parametric system,[7]

$$S = \langle X, U, f, x^{\mathrm{init}}, Y, h; \Theta \rangle.$$

Let $\hat{Y} \subseteq \mathbb{R}^{d_{\mathrm{target}}}$ be a space that we consider associated with the output space Y of S, and we call *target space*. Then,

- its *sample space* is $\mathcal{Z}^S := U^+ \times \hat{Y}$, and we define the following restricted variants, for $r \in \mathbb{R}^{\geq 0}$ and $T \in \mathbb{T}$:
  - the *T-horizon sample space* is $\mathcal{Z}^S_{\leq r} := U^{+\leq T} \times \hat{Y}$;
  - the *r-truncated sample space* is $\mathcal{Z}^S_{\leq r} := U^+_{\leq r} \times \hat{Y}_{\leq r}$;
  - the *T-horizon r-truncated sample space* is $\mathcal{Z}^{S,\leq T}_{\leq r} := U^{+\leq T}_{\leq r} \times \hat{Y}_{\leq r}$;
- the set of *admissible loss functions* $\mathfrak{L}^S$ consists of each analytic function $\ell : Y \times \hat{Y} \to \mathbb{R}$;
- the *objective function* induced by an admissible loss function $\ell \in \mathfrak{L}^S$ on a sample $z = (u, y) \in \mathcal{Z}^S$ is

$$\mathcal{L}^\ell_{S,z} : \Theta \to \mathbb{R}, \qquad \mathcal{L}^\ell_{S,z}(\theta) := \ell\big(\check{S}(u;\theta),\, y\big).$$

[7] We recall that Ω_Y denotes the Euclidean ambient space of Y.

When S is clear from the context, we write $\mathcal{Z}$ in place of $\mathcal{Z}^S$, we write $\mathcal{Z}_{\leq r}$ in place of $\mathcal{Z}^S_{\leq r}$, and we write $\mathcal{L}^\ell_z$ in place of $\mathcal{L}^\ell_{S,z}$; when also the loss function ℓ is clear from the context, we write $\mathcal{L}_z$ in place of $\mathcal{L}^\ell_{S,z}$; *End of Definition 189.*

Definition 190 (Loss-gradient growth)**.** Given a parametric Euclidean system S, a radius $r \in \mathbb{R}^{\geq 0}$, and an admissible loss function $\ell \in \mathfrak{L}^S$, the corresponding *loss-gradient growth function* is

$$\mathsf{G}^{\mathrm{S},\ell,\mathrm{r}}_{\nabla\mathcal{L}} : \mathbb{T} \to \mathbb{R}^{\geq 0}, \qquad \mathsf{G}^{\mathrm{S},\ell,\mathrm{r}}_{\nabla\mathcal{L}}(T) := \sup\left\{ \left\|\mathsf{D}\mathcal{L}^\ell_{S,z}(\theta)\right\| \;:\; z \in \mathcal{Z}^{S,\leq T}_{\leq r},\ \theta \in \Theta_{\leq r} \right\}.$$

End of Definition 190.

Convention 1 (Ambient extensions of admissible loss functions)**.** *Every admissible loss function $\ell : Y \times \hat{Y} \to \mathbb{R}$ admits an ambient extension $\bar{\ell}$ by Definition 38. The extension $\bar{\ell}$ is continuously-differentiable since it is analytic. Derivatives of ℓ are understood as the corresponding derivatives of $\bar{\ell}$ evaluated on $Y \times \hat{Y}$.*

Definition 191 (BIBLG)**.** A Euclidean parametric system S, having parameter space Θ and sample space $\mathcal{Z}^S$, satisfies *bounded-input bounded-loss-gradient (BIBLG)* if, for every admissible loss function $\ell \in \mathfrak{L}^S$ and every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for every $z \in \mathcal{Z}^S_{\leq r}$,

$$\mathcal{L}^\ell_{S,z} \text{ is differentiable on } \Theta_{\leq r} \qquad \text{and} \qquad \sup_{\theta \in \Theta_{\leq r}} \left\|\mathsf{D}\mathcal{L}^\ell_{S,z}(\theta)\right\| \leq B_r\,.$$

End of Definition 191.

Definition 192 (ae-BIBLG)**.** A measurable Euclidean parametric system S having parameter space Θ satisfies *almost-everywhere bounded-input bounded-loss-gradient (ae-BIBLG)* if for every admissible loss function $\ell \in \mathfrak{L}^S$ and every $r \in \mathbb{R}^{\geq 0}$, there exists $B_{\ell,r} \in \mathbb{R}^{\geq 0}$ such that, for every $z \in \mathcal{Z}^S_{\leq r}$, there exists

$$\text{a full-measure subset } \hat{\Theta}_{\leq r} \subseteq \Theta_{\leq r}$$

such that

$$\mathcal{L}^\ell_{S,z} \text{ is differentiable on } \hat{\Theta}_{\leq r} \qquad \text{and} \qquad \sup_{\theta \in \hat{\Theta}_{\leq r}} \left\|\mathsf{D}\mathcal{L}^\ell_{S,z}(\theta)\right\| \leq B_{\ell,r}\,.$$

End of Definition 192.

Definition 193 ($\tilde{\mathsf{D}}$-BIBLG)**.** A Euclidean parametric system S having parameter space Θ satisfies *bounded-input bounded-loss-gradient with respect to* $\tilde{\mathsf{D}}$, abbreviated as $\tilde{\mathsf{D}}$-BIBLG, if, for every loss function $\ell \in \mathfrak{L}^S$ and every $r \in \mathbb{R}^{\geq 0}$, there exists $B_{\ell,r} \in \mathbb{R}^{\geq 0}$ such that, for every sample $z \in \mathcal{Z}^S_{\leq r}$,

$$\tilde{\mathsf{D}} \text{ is applicable to } \mathcal{L}^\ell_{S,z}|_{\Theta_{\leq r}} \qquad \text{and} \qquad \sup_{\theta \in \Theta_{\leq r}} \left\|\mathsf{D}\mathcal{L}^\ell_{S,z}(\theta)\right\| \;\leq\; B_{\ell,r}\,.$$

End of Definition 193.

Definition 194 (Objective computation-graph family)**.** For S a Euclidean parametric system, an *objective computation-graph family* for S is a computation-graph family $\mathcal{G}$ for the family of loss functions $\mathcal{L}_S$. *End of Definition 194.*

Definition 195 (Graph-uniform selected BIBLG)**.** Let S be a Euclidean parametric system, let $\mathcal{G}$ be an objective computation-graph family for S, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator that is intensionally-admissible for $\mathcal{G}$.

We say that S satisfies *$\mathcal{G}$-uniform selected bounded-input bounded-loss-gradient ($\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG)* if, for every $\ell\mathfrak{L}^S$ and every $r \in \mathbb{R}^{\geq 0}$, there exists $B_{\ell,r} \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \left\|\tilde{\mathsf{D}}G(\theta)\right\| \leq B_{\ell,r} \qquad\qquad \forall z \in \mathcal{Z}^S_{\leq r},\ \forall t \in \mathbb{T},\ \forall G \in \mathcal{G}\left[\mathcal{L}^\ell_{S,z}\right]\,.$$

End of Definition 195.

Definition 196. A class $\mathcal{D}^{\mathrm{p}}$ of Euclidean parametric systems satisfies *BIBLG*, *ae-BIBLG*, *$\tilde{\mathsf{D}}$-BIBLG*, or *$\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG* if every system $S \in \mathcal{D}^{\mathrm{p}}$ satisfies BIBLG, ae-BIBLG, $\tilde{\mathsf{D}}$-BIBLG, and $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG, respectively. *End of Definition 196.*

F.5 nnVSD: Non-necessarily Vanishing State Derivative

Definition 197. Euclidean dynamics $D = \langle X, U, f \rangle$ satisfy the *non-necessarily vanishing state-derivative (nnVSD)* property if there exist a state $\dot{x} \in X$, a bounded sequence $u \in U^{\omega\mathrm{b}}$, and a positive constant $B \in \mathbb{R}^{>0}$ such that, for every $t \in \mathbb{T}$,

$$\check{D}_t^u \textit{ is differentiable at } \dot{x} \qquad \text{and} \qquad \left\| \mathsf{D}\check{D}_t^u(\dot{x}) \right\| \geq B \,.$$

Then,

- a Euclidean parametric system S^{p} satisfies *nnVSD* if its dynamics $\mathrm{dyn}(S^{\mathrm{p}})$ satisfy nnVSD;
- a class $\mathcal{D}^{\mathrm{p}}$ of Euclidean parametric dynamics satisfies *nnVSD* if some dynamics $D^{\mathrm{p}} \in \mathcal{D}^{\mathrm{p}}$ satisfy nnVSD;
- a class $\mathcal{S}^{\mathrm{p}}$ of Euclidean parametric systems satisfies *nnVSD* if some system $S^{\mathrm{p}} \in \mathcal{S}^{\mathrm{p}}$ satisfies nnVSD.

End of Definition 197.

G Architecture-agnostic Results: Expressivity, BIBS, BIBO, BIBSD, BIBLG

Table of Contents

G.1 Results for Expressivty 120
G.2 Results for BIBS and BIBO 122
G.3 Results for BIBSD and BIBLG 126

G.1 Results for Expressivty

G.1.1 η-finite expressivity implies positive-margin expressivity

We next relate η-expressivity to positive-margin expressivity. The key observation is that η-finiteness provides compact state and input spaces, while a continuous decoder into a finite alphabet separates the compact sets corresponding to distinct output symbols by a positive metric distance.

Lemma 7 (Positive margin of finite-valued decoders on compact sets)**.** *Let Y be a metric space, let Γ be a finite alphabet endowed with the discrete topology, let*

$$\mathrm{dec} : Y \to \Gamma$$

be continuous, and let $K \subseteq Y$ be compact.

Then there exists $\gamma \in \mathbb{R}^{>0}$ such that, for every $y \in K$ and $y' \in Y$,

$$d_Y(y, y') < \gamma \quad \Longrightarrow \quad \mathrm{dec}(y') = \mathrm{dec}(y). \tag{31}$$

End of Lemma 7.

Proof. For every $a \in \mathrm{dec}(K)$, define

$$K_a := K \cap \mathrm{dec}^{-1}(\{a\}).$$

Since Γ is discrete, the singleton $\{a\}$ is both open and closed. Since dec is continuous,

$$\mathrm{dec}^{-1}(\{a\})$$

is therefore both open and closed in Y. Consequently, K_a is compact and

$$Y \setminus \mathrm{dec}^{-1}(\{a\})$$

is closed.

If

$$Y \setminus \mathrm{dec}^{-1}(\{a\}) \neq \emptyset,$$

the compact set K_a and the closed set

$$Y \setminus \mathrm{dec}^{-1}(\{a\})$$

are disjoint. Hence their distance

$$\delta_a := \inf \left\{ d_Y(y, y') \;\middle|\; y \in K_a,\ y' \in Y \setminus \mathrm{dec}^{-1}(\{a\}) \right\}$$

is strictly positive. If instead

$$Y \setminus \mathrm{dec}^{-1}(\{a\}) = \emptyset,$$

define

$$\delta_a := 1.$$

Since $\mathrm{dec}(K)$ is a nonempty finite set,

$$\gamma := \min_{a \in \mathrm{dec}(K)} \delta_a$$

is strictly positive.

Let now $y \in K$ and $y' \in Y$ satisfy

$$d_Y(y, y') < \gamma.$$

Let

$$a := \mathrm{dec}(y).$$

Then $y \in K_a$. If $\mathrm{dec}(y') \neq a$, then

$$y' \in Y \setminus \mathrm{dec}^{-1}(\{a\}),$$

and hence

$$d_Y(y, y') \geq \delta_a \geq \gamma,$$

which is a contradiction. Therefore

$$\mathrm{dec}(y') = \mathrm{dec}(y),$$

proving Equation (31). □

Theorem 18 (η-finite recognition implies positive-margin recognition)**.** *Let*

$$S = \langle X, U, f, x_0, Y, h \rangle$$

be an η-finite system, let Σ be a finite alphabet, and suppose that S recognises a language L over Σ with continuous encoder

$$\mathrm{enc} : \Sigma \to U$$

and continuous decoder

$$\mathrm{dec} : Y \to \{0, 1\}.$$

Then S recognises L with positive margin, with the same encoder enc *and decoder* dec*.* End of Theorem 18.

Proof. Since S is η-finite, its state space X and input space U are finite unions of compact η-components. Hence X and U are compact.

Therefore,

$$X \times U \times X$$

is compact. Since the output function

$$h : X \times U \times X \to Y$$

is continuous, its image

$$K := h\big(X \times U \times X\big)$$

is compact.

In particular, the reachable output set

$$R := \big\{ S(\mathrm{enc}(w)) \mid w \in \Sigma^+ \big\}$$

satisfies

$$R \subseteq K.$$

By Lemma 7, there exists $\gamma \in \mathbb{R}^{>0}$ such that, for every $y \in K$ and $y' \in Y$,

$$d_Y(y, y') < \gamma \quad \Longrightarrow \quad \mathrm{dec}(y') = \mathrm{dec}(y).$$

Since $R \subseteq K$, the same implication holds for every $y \in R$. Therefore, dec admits a γ-margin on the reachable output set R.

Since S already recognises L with encoder enc and decoder dec, Definition 172 implies that S recognises L with margin γ. Since $\gamma > 0$, it recognises L with positive margin. □

Corollary 13 (η-expressivity implies positive-margin expressivity)**.** *Let $\mathcal{S}$ be a class of systems. Every language in the η-expressivity of $\mathcal{S}$ belongs to its positive-margin expressivity.* End of Corollary 13.

Proof. Let L belong to the η-expressivity of $\mathcal{S}$. By Definition 174, there exists an η-finite system $S \in \mathcal{S}$ that recognises L. By Theorem 18, the same system S recognises L with positive margin. □

G.2 Results for BIBS and BIBO

Definition 198 (Boundedness-preserving functions)**.** Let X and Y be metric spaces. A function $g : X \to Y$ is *boundedness-preserving* if $g(K)$ is bounded for every bounded set $K \subseteq X$.

End of Definition 198.

Lemma 8. *Let X and Y be metric spaces having Euclidean ambient spaces $\Omega_X = \mathbb{R}^m$ and $\Omega_Y = \mathbb{R}^n$. Suppose that X is closed in Ω_X. Then every continuous function $g : X \to Y$ is boundedness-preserving.*

Proof. Let $K \subseteq X$ be bounded. Since X carries the metric induced by Ω_X, K is also bounded as a subset of Ω_X. Since X is closed in Ω_X, the closure $K^\uparrow$ of K in Ω_X is contained in X. Furthermore, $K^\uparrow$ is closed and bounded in the Euclidean space Ω_X, and hence compact. Since g is continuous, $g(K^\uparrow)$ is compact and therefore bounded. Since $g(K) \subseteq g(K^\uparrow)$, we conclude that $g(K)$ is bounded. Thus g is boundedness-preserving. □

Proposition 28 (Boundedness preservation of analytic Euclidean maps)**.** *Let $f : X \times \Theta \to Y$ be a parametric analytic Euclidean map. For every $\theta \in \Theta$, the function*

$$f[\theta] : X \to Y \qquad\qquad f[\theta](x) = f(x; \theta)$$

is boundedness-preserving.

Proof. Fix $\theta \in \Theta$, and let $f^\uparrow$ be an analytic ambient extension of f. Let $K \subseteq X$ be bounded. Since X carries the metric induced by Ω_X, K is bounded in Ω_X. Hence its closure $K^\uparrow$ in Ω_X is compact. The function $x \mapsto f^\uparrow(x; \theta)$ is continuous, and hence $f^\uparrow(K^\uparrow, \theta)$ is compact and hence bounded. Since $f^\theta(K) \subseteq f^\uparrow(K^\uparrow, \theta)$, we conclude that $f^\theta(K)$ is bounded. Thus f^θ is boundedness-preserving. □

Corollary 14. *Let $f : X \times \Theta \to Y$ be a parametric analytic Euclidean map, let $\theta \in \Theta$, let $f_\theta := f[\theta]$, and let $\hat{X} \subseteq X$. Then, the restriction $f_\theta|_{\hat{X}}$ is boundedness-preserving.*

Proof. Let $K \subseteq \hat{X}$ be bounded. Then K is a bounded subset of X. By Proposition 28, $f(\theta; K)$ is bounded. Hence the restriction $f[\theta]|_{\hat{X}}$ is boundedness-preserving. □

Corollary 15. *Let $f : X \times \to Y$ be an analytic Euclidean map, and let $\hat{X} \subseteq X$. Then, the restriction $f|_{\hat{X}}$ is boundedness-preserving.*

Proof. Let us define the parametric Euclidean function

$$f^{\mathrm{p}} : X \times \{\theta_0\} \to Y\,, \qquad\qquad f^{\mathrm{p}}(x; \theta_0) := f(x)\,.$$

We have that f^{p} is analytic, and hence, by Corollary 14, we have that $f^{\mathrm{p}}[\theta_0]|_{\hat{X}}$ is boundedness-preserving. Therefore f is boundedness-preserving since $f = f^{\mathrm{p}}[\theta_0]$. □

Lemma 9 (BIBS implies BIBO)**.** *Let S be a Euclidean system whose output function h is boundedness-preserving. If S satisfies BIBS, then S satisfies BIBO.*

Proof. Let $x_0 \in X$ and $u = (u_t)_{t \in \mathbb{T}} \in U^{\omega\mathrm{b}}$ be arbitrary.

Let $(x_t)_{t \in \mathbb{N}}$ be the state sequence of the dynamics $D := \mathrm{dyn}(S)$ on input u when started in x_0; namely,

$$x_t = \check{D}(x_0, u_{[1..t]}) \qquad \forall t \in \mathbb{T}\,.$$

Since S satisfies BIBS, there exists $B_X \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t \in \mathbb{N}} \|x_t\| \leq B_X\,.$$

Thus, the state sequence $(x_t)_{t \in \mathbb{N}}$ is bounded. Furthermore, u is bounded by assumption. Hence the set

$$K := \big\{(x_{t-1}, u_t, x_t) \,:\, t \in \mathbb{T}\big\}$$

is bounded in $X \times U \times X$.

Let $(y_t)_{t\in\mathbb{T}}$ be the output sequence of the S on input u; namely,

$$y_t = \check{S}(u_{[1..t]}) \qquad \forall t \in \mathbb{T}\,.$$

Since h is boundedness-preserving, we have that $h(K)$ is bounded. Therefore, there exists $B_Y \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \|y_t\| \leq B_Y\,.$$

Thus S satisfies BIBO. □

Corollary 16 (Analytic output functions)**.** *Let S be a Euclidean parametric system whose output function h is analytic. If S satisfies wBIBS, then S satisfies wBIBO.*

Proof. Let

$$S = \langle X, U, f, x^{\mathrm{init}}, Y, h; \Theta^f, \Theta^h\rangle,$$

and $(\theta^f, \theta^h) \in \Theta^f \times \Theta^h$ be arbitrary.

Since S satisfies wBIBS, we have that $S[\theta^f, \theta^h]$ satisfies BIBS.

By Corollary 14, the output function $h[\theta_h]$ is boundedness-preserving.

By Lemma 9 it follows that $S[\theta^f, \theta^h]$ satisfies BIBO, Since (θ^f, θ^h) was arbitrary, we conclude that S satisfies wBIBO. □

Lemma 10 (BIBS under parallel composition)**.** *Let $D = D_1 \parallel \cdots \parallel D_n$ be a parallel composition of Euclidean dynamics. If every D_i satisfies BIBS, then D satisfies BIBS.*

Proof. Let $x_0 = \langle x_0^1, \ldots, x_0^n\rangle$ and $u = (u_t)_{t\in\mathbb{T}} \in U^{\omega\mathrm{b}}$ be arbitrary.

For every $i \in [1..n]$, let $(x_t^i)_{t\in\mathbb{N}}$ be the state sequence of D_i on input u when started in x_0^i; in particular,

$$x_t = \check{D}_i(x_0^i, u_{[1..t]}) \qquad \forall t \in \mathbb{T}\,;$$

since D_i satisfies BIBS, there exists $B_i \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \|x_t^i\| \leq B_i\,,$$

By the definition of parallel composition, we have that the state sequence $(x_t)_{t\in\mathbb{N}}$ of D on input u when started in x_0 is

$$x_t = \langle x_t^1, \ldots, x_t^n\rangle \qquad \forall t \in \mathbb{N}\,,$$

and hence

$$\|x_t\| \;=\; \sqrt{\sum_{i=1}^{n} \|x_t^i\|^2} \;\leq\; \sqrt{\sum_{i=1}^{n} B_i^2} \qquad\qquad \forall t \in \mathbb{N}\,.$$

Therefore D satisfies BIBS. □

Corollary 17 (Parametric wBIBS under parallel composition)**.** *Let $D^{\mathrm{p}} = D_1^{\mathrm{p}} \parallel \cdots \parallel D_n^{\mathrm{p}}$ be a parallel composition of parametric Euclidean dynamics. If every D_i satisfies wBIBS, then D satisfies wBIBS.*

Proof. Let $\theta = \langle \theta_1, \ldots, \theta_n\rangle$ be arbitrary. By Corollary 6,

$$D^{\mathrm{p}}[\theta] = D_1^{\mathrm{p}}[\theta_1] \parallel \cdots \parallel D_n^{\mathrm{p}}[\theta_n].$$

For every $i \in [1..n]$, since D_i^{p} satisfies wBIBS, its instance $D_i^{\mathrm{p}}[\theta_i]$ satisfies BIBS. By Lemma 10, it follows that $D^{\mathrm{p}}[\theta]$ satisfies BIBS. Since θ was arbitrary, we conclude that D^{p} satisfies wBIBS. □

Lemma 11 (BIBS of recurrent cascade dynamics)**.** *Let $\mathcal{D}$ be a class of Euclidean dynamics satisfying BIBS, let $\mathcal{F}$ be a class of boundedness-preserving functions, and let D be recurrent cascade dynamics of type $(\mathcal{D}, \mathcal{F})$. Then, dynamics D satisfy BIBS.* End of Lemma 11.

Proof. Let X be the state space of D, and let U be the state space of D.

We have that D admits a recurrent cascade presentation

$$\mathsf{C} \;=\; g_1 \blacktriangleright \mathsf{L}_1 \blacktriangleleft_{g_2} \mathsf{L}_2 \cdots \blacktriangleleft_{g_\ell} \mathsf{L}_\ell$$

where every g_i is an analytic Euclidean map, and every L_i is a recurrent layer dynamics presentation

$$\mathsf{L}_i \;=\; D_{i,1} \parallel \cdots \parallel D_{i,n_i}$$

with every unit D_{i,n_i} belonging to $\mathcal{D}$.

Since $\mathcal{D}$ satisfies BIBS, for every $i \in [1..\ell]$ and every $j \in [1..n_i]$, we have that $D_{i,j}$ satisfies BIBS. Hence, for every $i \in [1..\ell]$, since

$$D_i := [\![\mathsf{L}_i]\!] \;=\; D_{i,1} \parallel \cdots \parallel D_{i,n_i}\,,$$

by Lemma 10, we have that D_i satisfies BIBS.

Let $x_0 = \langle x_0^1, \ldots, x_0^\ell \rangle \in X$ and $u = (u_t)_{t\in\mathbb{T}} \in U^{\omega\mathrm{b}}$ be arbitrary.

For every $i \in [1..\ell]$ let $u^i := (u_t^i)_{t\in\mathbb{T}}$ be the sequence of inputs received by the layer D_i when D is executed on input u starting from state x_0, and let $x^i := (x_t^i)_{t\in\mathbb{T}}$ be the state sequence of D_i on input u^i when started in state x_0^i.

We proceed by induction on i from 1 to ℓ, showing that the sequences u^i and x^i are bounded; namely, we show there exist $B_1, \ldots, B_i \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\left\{\|u_t^k\|,\, \|x_t^k\|\right\} \;\leq\; B_k \qquad\qquad \forall k \in [1..i]\,.$$

In the base case we have $i = 1$. We have

$$u_t^1 \;=\; g_1(u_t) \qquad \forall t \in \mathbb{T}\,,$$

and hence the sequence u^1 is bounded, since the input sequence u is bounded and the function g_1 is analytic and hence boundedness-preserving by Corollary 15. Since D_1 satisfies BIBS, it follows that its state sequence x^1 is bounded. Thus, there exists $B_1 \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\left\{\|u_t^1\|,\, \|x_t^1\|\right\} \;\leq\; B_1\,.$$

In the inductive case we have $i \geq 2$, and we assume by induction that there exist $B_1, \ldots, B_{i-1} \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\left\{\|u_t^k\|,\, \|x_t^k\|\right\} \;\leq\; B_k \qquad\qquad \forall k \in [1..i-1]\,.$$

By the definition of cascade composition,

$$u_t^i \;=\; g_i\big(x_{t-1}^{i-1},\, u_{i,t}^{i-1},\, x_t^{i-1}\big) \qquad\qquad \forall t \in \mathbb{T}\,,$$

and hence the sequence u^i is bounded, since the sequences x^{i-1} and u^{i-1} are bounded by induction and the function g_i is analytic and hence boundedness-preserving by Corollary 15; since D_i satisfies BIBS, it follows that its state sequence x^i is bounded. Thus, there exists $B_i \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\left\{\|u_t^i\|,\, \|x_t^i\|\right\} \;\leq\; B_i\,,$$

and hence we have $B_1, \ldots, B_i \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\left\{\|u_t^k\|,\, \|x_t^k\|\right\} \;\leq\; B_k \qquad\qquad \forall k \in [1..i]\,.$$

We conclude that the state sequence $(x_t)_{t\in\mathbb{T}}$ of D on input u when started in x_0 is bounded since its components are components of the sequences $(x_t^i)_{t\in\mathbb{T}}$ for $i \in [1..\ell]$.

Since x_0 and u were arbitrary, we conclude that D satisfies BIBS. □

Theorem 19 (BIBS and BIBO of Recurrent Neural Cascades)**.** *Let $\mathcal{D}$ be a class of Euclidean dynamics satisfying BIBS.*

Then:

- *every $\mathcal{D}$-RNC dynamics satisfies BIBS;*
- *every $\mathcal{D}$-RNC satisfies BIBS and BIBO.*

Consequently, the class of $\mathcal{D}$-RNC dynamics satisfies BIBS, and the class of $\mathcal{D}$-RNCs satisfies BIBS and BIBO. End of Theorem 19.

Proof. Let D be arbitrary $\mathcal{D}$-RNC dynamics. By definition of RNC, we have that D are recurrent cascade dynamics of type $(\mathcal{D}, \mathcal{F}_{\text{AN}})$. Thus, they admit a recurrent cascade dynamics presentation where

- every unit of belongs to $\mathcal{D}$ and therefore satisfies BIBS;
- every connecting function is an analytic Euclidean map.

Every analytic Euclidean map is boundedness-preserving by Corollary 15, and hence, by Lemma 11, we conclude that implies that D satisfies BIBS.

Now let S be an arbitrary $\mathcal{D}$-RNC. By definition, its dynamics $\mathrm{dyn}(S)$ are $\mathcal{D}$-RNC dynamics, and its output function h is analytic.

By the first part of the proof, $\mathrm{dyn}(S)$ satisfies BIBS, and hence S satisfies BIBS. Furthermore, its output function h is boundedness-preserving by Corollary 15. Therefore, by Lemma 9, we conclude that S satisfies BIBO.

Since D and S were arbitrary, the corresponding classes satisfy the claimed properties. □

G.2.1 Results for uBIBS and uBIBO

The BIBS closure results for parallel and recurrent-cascade composition admit uniform parametric analogues.

Lemma 12 (uBIBS under parallel composition)**.** *Let D^{p} be parametric Euclidean dynamics given by a parallel composition $D^{\mathrm{p}} = D_1^{\mathrm{p}} \parallel \cdots \parallel D_n^{\mathrm{p}}$. If every D_i^{p} satisfies uBIBS, then D^{p} satisfies uBIBS.*

Proof. Let $D^{\mathrm{p}} = \langle X, U, f; \Theta \rangle$. Fix $r \in \mathbb{R}^{\geq 0}$ arbitrary. Let

$$\theta = \langle \theta_1, \ldots, \theta_n \rangle \in \Theta_{\leq r}\,, \qquad x_0 = \langle x_0^1, \ldots, x_0^n \rangle \in X_{\leq r}\,, \qquad u \in U_{\leq r}^{\omega}$$

be arbitrary. We have

$$\|\theta_i\| \leq r \qquad \text{and} \qquad \|x_0^i\| \leq r \qquad\qquad \forall i \in [1..n].$$

For every $i \in [1..n]$, since D_i satisfies uBIBS, there exists $B_{i,r} \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t \in \mathbb{T}} \|x_t^i\| \leq B_{i,r}\,.$$

The complete state is

$$x_t = \langle x_t^1, \ldots, x_t^n \rangle,$$

and hence

$$\|x_t\| \leq \sqrt{\sum_{i=1}^{n} B_{i,r}^2} \qquad \forall t \in \mathbb{T}.$$

Since r, θ, x_0, and u were arbitrary, we conclude that D satisfies uBIBS. □

Theorem 20 (uBIBS of parametric RNC dynamics)**.** *Let $\mathcal{D}^{\mathrm{p}}$ be a class of parametric Euclidean dynamics satisfying uBIBS. Then every parametric RNC dynamics in $\mathrm{RNC}_{\mathrm{dyn}}^{\mathrm{p}}(\mathcal{D}^{\mathrm{p}})$ satisfies uBIBS. Consequently, the class $\mathrm{RNC}_{\mathrm{dyn}}^{\mathrm{p}}(\mathcal{D}^{\mathrm{p}})$ satisfies uBIBS.*

Proof. Let $D \in \mathrm{RNC}^{\mathrm{p}}_{\mathrm{dyn}}(\mathcal{D}^{\mathrm{p}})$ be arbitrary. We have that D is given by a cascade composition

$$D \;=\; g_1 \triangleright D_1 \ltimes_{g_2} D_2 \cdots \ltimes_{g_L} D_L \,.$$

Every D_ℓ is given by a finite parallel composition of dynamics belonging to $\mathcal{D}^{\mathrm{p}}$, and every function g_ℓ is a parametric analytic Euclidean map. Thus, by Lemma 12, every D_ℓ satisfies uBIBS.

Fix $r \in \mathbb{R}^{\geq 0}$ arbitrary, and let

$$\theta \in \Theta_{\leq r}, \qquad x_0 \in X_{\leq r}, \qquad u \in U^{\omega}_{\leq r}$$

be arbitrary.

For every $\ell \in [1..L]$, let $u^\ell := (u^\ell_t)_{t\in\mathbb{T}}$ be the input sequence received by D_ℓ when D is executed on input u and started in x_0, and and let $(x^\ell_t)_{t\in\mathbb{T}}$ be the state sequence of D_ℓ on input u^ℓ when started in x^ℓ_0.

We proceed by induction on ℓ from 1 to L, showing there exist $B_1, \ldots, B_\ell \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\{u^\ell_t, x^\ell_t\} \;\leq\; B_i \qquad\qquad \forall i \in [1..\ell] \,.$$

In the base case we have $\ell = 1$. We have that the initial state x^1_0 and the parameter values of D_1 from the joint parameter values θ are bounded by r; furthermore,

$$u^1_t = g_1(u_t) \qquad\qquad \forall t \in \mathbb{T} \,,$$

and hence u^1 is bounded, since the input sequence u is bounded by r and g_1 is analytic and hence boundedness-preserving by Corollary 15. Since D_1 satisfies uBIBS, we conclude there exists $B_1 \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\{u^1_t, x^1_t\} \;\leq\; B_1 \,.$$

In the inductive case we have $\ell \geq 2$, and, by induction, there exist $B_1, \ldots, B_{\ell-1} \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\{u^i_t, x^i_t\} \;\leq\; B_i \qquad\qquad \forall i \in [1..\ell-1] \,.$$

The initial state x^ℓ_0 and the parameter values of D_ℓ from the joint parameter values θ are bounded by r; furthermore,

$$u^\ell_t \;=\; \left(x^\ell_{t-1},\, u^\ell_t,\, x^\ell_t\right) \qquad\qquad \forall t \in \mathbb{T} \,,$$

and hence u^ℓ is bounded, since $x^{\ell-1}, u^{\ell-1}$ are bounded by $B_{\ell-1}$ by the inductive hypothesis, and g_ℓ is analytic and hence boundedness-preserving by Corollary 15. Since D_ℓ satisfies uBIBS, we conclude there exists $B_\ell \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\{u^\ell_t, x^\ell_t\} \;\leq\; B_\ell \,.$$

Thus, by the inductive hypothesis, there exist $B_1, \ldots, B_\ell \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{t\in\mathbb{T}} \max\{u^i_t, x^i_t\} \;\leq\; B_i \qquad\qquad \forall i \in [1..\ell] \,.$$

Since the cascade decomposition of D given by dynamics D_i and functions g_i was arbitrary, and since r, θ, x_0, and u were arbitrary, we conclude that D satisfies uBIBS. In turn, since D was arbitrary, we conclude that the corresponding class $\mathrm{RNC}^{\mathrm{p}}_{\mathrm{dyn}}(\mathcal{D}^{\mathrm{p}})$ satisfies uBIBS. □

G.3 Results for BIBSD and BIBLG

G.3.1 Auxiliary results

Lemma 13 (Graph-uniform selected-derivative bounds under analytic extensions)**.** *Let $h : X \times \Phi \to Y$ be an analytic parametric Euclidean map.*

Let Λ be a Euclidean parameter space.

Let $A \subseteq \mathbb{N}^{>0}$ be an arbitrary index set. For every $a \in A$, let $p_a : \Lambda \to X$ and $q_a : \Lambda \to \Phi$ be functions, and let $\mathcal{P}_a$ and $\mathcal{Q}_a$ be indexed computation-graph–families for p_a and q_a, respectively.

For every $a \in A$, we define the function

$$h_a(\lambda) \;:=\; h\big(p_a(\lambda);\; q_a(\lambda)\big),$$

and the the computation-graph–family

$$\mathcal{H}_a \;:=\; \Big\{\, (P_a, Q_a) \triangleright h \;\Big|\; P_a \in \mathcal{P}[a]\,,\; Q_a \in \mathcal{Q}[a] \,\Big\}\,.$$

Let $\tilde{\mathsf{D}}$ be a pseudoderivative operator such that, for every $a \in A$, it is admissible for the graph families $\mathcal{P}_a, \mathcal{Q}_a$, and $\mathcal{H}_a$.

Let $r \in \mathbb{R}^{\geq 0}$, and suppose there exist $B_r^q, \hat{B}_r^q, B_r^p, \hat{B}_r^p \in \mathbb{R}^{\geq 0}$ such that, for every $a \in A$,

$$\sup_{\lambda \in \Lambda_{\leq r}} \|q_a(\lambda)\| \leq B_r^q\,, \qquad \sup_{Q_a \in \mathcal{Q}_a} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} Q_a(\lambda)\| \leq \hat{B}_r^q\,,$$

$$\sup_{\lambda \in \Lambda_{\leq r}} \|p_a(\lambda)\| \leq B_r^p\,, \qquad \sup_{P_a \in \mathcal{P}_a} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} P_a(\lambda)\| \leq \hat{B}_r^p\,.$$

Then, there exist $C_r, \hat{C}_r \in \mathbb{R}^{\geq 0}$ such that, for every $a \in A$,

(1) every graph $H_a \in \mathcal{H}_a$ represents h_a;

(2) the two following inequalities hold:

$$\sup_{\lambda \in \Lambda_{\leq r}} \|h_a(\lambda)\| \leq C_r\,, \qquad \sup_{H_a \in \mathcal{H}_a} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} H_a(\lambda)\| \leq \hat{C}_r\,.$$

Proof. Let $\rho_r := \max(B_r^p, B_r^q)$. By Proposition 28, there exist $M_r, M_r^p, M_r^q \in \mathbb{R}^{\geq 0}$ such that, whenever $x \in X_{\leq \rho_r}$ and $\phi \in \Phi_{\leq \rho_r}$, we have

$$\|h(x;\phi)\| \leq M_r\,, \qquad \big\|\mathsf{D}_x h(x;\phi)\big\| \leq M_r^p\,, \qquad \big\|\mathsf{D}_\phi h(x;\phi)\big\| \leq M_r^q\,.$$

Let $a \in A$ be arbitrary. For every $\lambda \in \Lambda_{\leq r}$, the assumed value bounds give

$$p_a(\lambda) \in X_{\leq \rho_r}\,, \qquad q_a(\lambda) \in \Phi_{\leq \rho_r}\,.$$

Consequently,

$$\|h_a(\lambda)\| \;=\; \big\|h\big(p_a(\lambda); q_a(\lambda)\big)\big\| \;\leq\; M_r\,.$$

Hence

$$\sup_{\lambda \in \Lambda_{\leq r}} \|h_a(\lambda)\| \;\leq\; M_r\,.$$

Now let

$$H_a := (P_a, Q_a) \triangleright h \in \mathcal{H}_a$$

be arbitrary. By Proposition 26, the computation graph H_a represents the function

$$\lambda \mapsto h\big(p_a(\lambda); q_a(\lambda)\big) \;=\; h_a(\lambda)\,.$$

This proves the first claim.

By Proposition 27, for every $\lambda \in \Lambda_{\leq r}$,

$$\tilde{\mathsf{D}} H_a(\lambda) \;=\; \Big(\mathsf{D}_x h\big(p_a(\lambda); q_a(\lambda)\big) \cdot \tilde{\mathsf{D}} P_a(\lambda)\Big) + \Big(\mathsf{D}_\phi h\big(p_a(\lambda); q_a(\lambda)\big) \cdot \tilde{\mathsf{D}} Q_a(\lambda)\Big)\,.$$

Therefore,

$$\begin{aligned}\|\tilde{\mathsf{D}} H_a(\lambda)\| \;&\leq\; M_r^p \cdot \|\tilde{\mathsf{D}} P_a(\lambda)\| + M_r^q \cdot \|\tilde{\mathsf{D}} Q_a(\lambda)\| \\ &\leq\; M_r^p \cdot \hat{B}_r^p + M_r^q \cdot \hat{B}_r^q\,.\end{aligned}$$

Since $P_a \in \mathcal{P}_a$ and $Q_a \in \mathcal{Q}_a$ were arbitrary, we obtain

$$\sup_{H_a \in \mathcal{H}_a} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} H_a(\lambda)\| \;\leq\; M_r^p \cdot \hat{B}_r^p + M_r^q \cdot \hat{B}_r^q \,.$$

Since $a \in A$ was arbitrary and all the constants above are independent of a, the claim follows by taking

$$C_r \;:=\; M_r, \qquad \hat{C}_r \;:=\; M_r^p \cdot \hat{B}_r^p + M_r^q \cdot \hat{B}_r^q \,.$$

□

Corollary 18 (Coordinate-projection case)**.** *Assume the setting of Lemma 13.*

Suppose $q : \Lambda \to \Phi$ is a linear map satisfying

$$\textit{(I)} \quad \|q(\lambda)\| \leq \|\lambda\| \quad \forall\, \lambda \in \Lambda \,, \qquad\qquad \textit{(II)} \quad \|q(\lambda)\| q_a = q \quad \forall\, a \in A \,, \tag{32}$$

Suppose Q is a computation graph representing q and satisfying

$$\textit{(III)} \quad \tilde{\mathsf{D}} Q(\lambda) = \mathsf{D} q(\lambda) \quad \forall\, \lambda \in \Lambda \,, \qquad\qquad \textit{(IV)} \quad \mathcal{Q}[a] = \{Q\} \quad \forall\, a \in A \,. \tag{33}$$

Let $r \in \mathbb{R}^{\geq 0}$, and suppose that there exist $B_r^p, \hat{B}_r^p \in \mathbb{R}^{\geq 0}$ such that, for every $a \in A$,

$$\sup_{\lambda \in \Lambda_{\leq r}} \|p_a(\lambda)\| \leq B_r^p \,, \qquad\qquad \sup_{P_a \in \mathcal{P}[a]} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} P_a(\lambda)\| \leq \hat{B}_r^p \,.$$

Then, there exist $C_r, \hat{C}_r \in \mathbb{R}^{\geq 0}$ such that, for every $a \in A$,

$$\sup_{\lambda \in \Lambda_{\leq r}} \|h_a(\lambda)\| \leq C_r \,, \qquad\qquad \sup_{H_a \in \mathcal{H}_a} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} H_a(\lambda)\| \leq \hat{C}_r \,.$$

Proof. By condition (I) in Equation (32),

$$\sup_{\lambda \in \Lambda_{\leq r}} \|q_a(\lambda)\| = \sup_{\lambda \in \Lambda_{\leq r}} \|q(\lambda)\| \leq r \qquad \forall\, a \in A.$$

Furthermore, since q is linear, its derivative is constant and satisfies

$$\|\mathsf{D} q(\lambda)\| = \|q\|_{\mathrm{op}} \leq 1 \qquad \forall\, \lambda \in \Lambda,$$

where the final inequality follows from condition (I) in Equation (32). Hence, by the condition (III) in Equation (33),

$$\sup_{Q_a \in \mathcal{Q}[a]} \sup_{\lambda \in \Lambda_{\leq r}} \|\tilde{\mathsf{D}} Q_a(\lambda)\| \leq 1 \qquad \forall\, a \in A.$$

Thus, the hypotheses of Lemma 13 hold with

$$B_r^q := r, \qquad \hat{B}_r^q := 1.$$

The claim follows. □

Definition 199 (Output-context parameter function)**.** Let $D = \langle X, U, f; \Theta \rangle$ be Euclidean parametric dynamics, let $x_0 \in X$, and let $u = u_1 u_2 \cdots u_t \in U^+$.

The *output-context parameter function* for (x_0, u) is the function $\gamma_u : \Theta \to (X \times U \times X)$ given by

$$\gamma_u(\theta) := (x_{t-1}, u_t, x_t) \qquad \text{for} \qquad x_{t-1} := \check{D}(x_0, u_{[1..t-1]}; \theta), \;\; x_t := \check{D}(x_0, u; \theta).$$

End of Definition 199.

Lemma 14 (Uniform state bounds imply uniform output-context bounds)**.** *Let $D = \langle X, U, f; \Theta \rangle$ be Euclidean parametric dynamics satisfying uBIBS, let $x_0 \in X$, and let γ_u be the output-context parameter function of Definition 199.*

Then, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \left\|\gamma_u(\theta)\right\| \leq B_r \qquad\qquad \forall u \in U^+_{\leq r} \,.$$

Proof. Fix $r \in \mathbb{R}^{\geq 0}$, and let $\rho_r := \max(r, \|x_0\|)$. Let $u = u_1 \cdots u_t \in U^+_{\leq r}$ be arbitrary, and extend it to $\bar{u} \in U^\omega_{\leq \rho_r}$ by repeating u_t after time t.

By uBIBS, applied at radius ρ_r, there exists $M_{\rho_r} \in \mathbb{R}^{\geq 0}$, independent of u, such that, for every $\theta \in \Theta_{\leq r} \subseteq \Theta_{\leq \rho_r}$, every state on input $\bar{u}$ has norm at most M_{ρ_r}.

Let

$$\bar{M}_r := \max\left\{M_{\rho_r}, \|x_0\|\right\}.$$

Both state components of $\gamma_u(\theta)$ therefore have norm at most $\bar{M}_r$, and additionally $\|u_t\| \leq r$. Hence

$$\sup_{\theta \in \Theta_{\leq r}} \left\|\gamma_u(\theta)\right\| \;\leq\; \sqrt{2 \cdot \bar{M}_r^2 + r^2}\,.$$

Since the right-hand side is independent of u, the claim follows. □

G.3.2 BIBSD: Graph-Uniform selected-derivative bounds for state gradients

Recall the following notions: PAP functions and their intensional derivatives computation graph (in §A.12); computation graphs and selected intensional derivatives (in §E); the relevant BIBSD properties relative to selected derivative operators and and the following BIBSD variants: ae-uBIBSD (Definition 183), selected-uniform–uBIBSD (Definition 184), and graph-uniform–uBIBSD (Definition 186).

Proposition 29 (Graph-uniform bounds imply selected-uniform–uBIBSD)**.** *Let us consider parametric Euclidean dynamics $D = \langle X, U, f; \Theta\rangle$ and their parameter-to-state function family $\check{D}$. Let $\mathcal{G}$ be a computation-graph family for $\check{D}$, and let Γ be a graph selector for $\check{D}$ over $\mathcal{G}$, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator such that D satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD. Then,*

(1) the selected intensional derivative operator $\tilde{\mathsf{D}}^\Gamma$ is applicable to the family $\check{D}$;

(2) the dynamics D satisfy $\tilde{\mathsf{D}}^\Gamma$-uBIBSD.

Proof. Since D satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD, Definition 186 implies, in particular, that $\tilde{\mathsf{D}}$ is intensionally-admissible for every computation graph in $\mathcal{G}$.

Let $q \in \check{D}$ be arbitrary. Since Γ is a graph selector for $\check{D}$ over $\mathcal{G}$, we have $\Gamma(q) \in \mathcal{G}[q]$ by Definition 158, and hence $\tilde{\mathsf{D}}$ is intensionally-admissible for $\Gamma(q)$. Since $\Gamma(q)$ represents q, Proposition 24 implies that q is PAP, and also that $\tilde{\mathsf{D}}\,\Gamma(q) \in \partial_\bullet q$.[8] By Definition 164, we have $\tilde{\mathsf{D}}^\Gamma q = \tilde{\mathsf{D}}\,\Gamma(q)$, and hence $\tilde{\mathsf{D}}^\Gamma q \in \partial_\bullet q$. Since $q \in \check{D}$ was arbitrary, $\tilde{\mathsf{D}}^\Gamma$ is applicable to $\check{D}$ and it is a selected intensional derivative operator for this family.

It remains to establish the uniform bound. Let $r \in \mathbb{R}^{\geq 0}$ be arbitrary. Since D satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \left\|\tilde{\mathsf{D}}G(\theta)\right\| \leq B_r \qquad \forall u \in U^\omega_{\leq r},\;\; \forall x_0 \in X_{\leq r},\;\; \forall t \in \mathbb{T},\;\; \forall G \in \mathcal{G}\Big[\check{D}[u, x_0, t]\Big]\,.$$

In particular, since Γ is a graph selector,

$$\Gamma\big(\check{D}[u, x_0, t]\big) \in \mathcal{G}\Big[\check{D}[u, x_0, t]\Big]\,.$$

Hence

$$\sup_{\theta \in \Theta_{\leq r}} \Big\|\tilde{\mathsf{D}}\Gamma\big(\check{D}[u, x_0, t]\big)(\theta)\Big\| \leq B_r\,.$$

By Definition 164, the above is equivalent to

$$\sup_{\theta \in \Theta_{\leq r}} \Big\|\tilde{\mathsf{D}}^\Gamma \check{D}[u, x_0, t](\theta)\Big\| \leq B_r\,.$$

Since $r \in \mathbb{R}^{\geq 0}$ was arbitrary, Definition 184 implies that D satisfies $\tilde{\mathsf{D}}^\Gamma$-uBIBSD. □

[8] Recall that the function $\tilde{\mathsf{D}}\,\Gamma(q) \in \partial_\bullet q$ means that $\tilde{\mathsf{D}}\,\Gamma(q)$ is an intensional derivative function of q. See §A.12.

Proposition 30 (Graph-uniform state-derivative bounds imply ae-uBIBSD under a.e correctness)**.** *Let us consider measurable-Euclidean parametric dynamics $D = \langle X, U, f; \Theta \rangle$ and their parameter-to-state function family $\check{D}$. Let $\mathcal{G}$ be an indexed computation-graph family for $\check{D}$, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator such that D satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$–uBIBSD.*

Suppose that, for every $q \in \check{D}$, every $G \in \mathcal{G}[q]$, and every $r \in \mathbb{R}^{\geq 0}$, there exists a full-measure subset $\hat{\Theta}_r^{q,G} \subseteq \Theta_{\leq r}$ such that, for all parameter values $\theta \in \hat{\Theta}_r^{q,G}$,

$$q \text{ is differentiable at } \theta \qquad \text{and} \qquad \mathsf{D}q(\theta) = \tilde{\mathsf{D}}G(\theta)\,.$$

Then D satisfies ae-uBIBSD.

Proof. Let $r \in \mathbb{R}^{\geq 0}$ be arbitrary. Since D satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-uBIBSD, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \left\| \tilde{\mathsf{D}}G(\theta) \right\| \leq B_r \qquad \forall\, u \in U_{\leq r}^{\omega},\ \forall\, x_0 \in X_{\leq r},\ \forall\, t \in \mathbb{T},\ \forall\, G \in \mathcal{G}\Big[\check{D}[x_0, u, t]\Big]\,. \tag{34}$$

Now, let $u \in U_{\leq r}^{\omega}$ and $x_0 \in X_{\leq r}$ be arbitrary, and let

$$q_t := \check{D}[x_0, u, t] \ \in\ \check{D} \qquad \forall\, t \in \mathbb{T}\,.$$

Since $\mathcal{G}$ is an indexed computation-graph family for $\check{D}$, for every $t \in \mathbb{T}$, there exists a computation graph

$$G_t \in \mathcal{G}[q_t]\,.$$

By the assumed a.e. correctness, for every $t \in \mathbb{T}$, there exists a full-measure subset

$$\hat{\Theta}_r^{q_t, G_t} \subseteq \Theta_{\leq r}$$

such that, for every $\theta \in \hat{\Theta}_r^{q_t, G_t}$,

$$q_t \text{ is differentiable at } \theta \qquad \text{and} \qquad \mathsf{D}q_t(\theta) = \tilde{\mathsf{D}}G_t(\theta)\,.$$

Let us define

$$\hat{\Theta}_{r,u,x_0} := \bigcap_{t \in \mathbb{T}} \hat{\Theta}_r^{q_t, G_t}\,.$$

Since $\mathbb{T}$ is countable, we have that $\hat{\Theta}_{r,u,x_0}$ is a full-measure subset of $\Theta_{\leq r}$.

Let $\theta \in \hat{\Theta}_{r,u,x_0}$ and $t \in \mathbb{T}$ be arbitrary. Then $\theta \in \hat{\Theta}_r^{q_t, G_t}$, and hence $\check{D}[x_0, u, t] = q_t$ is differentiable at θ. Furthermore, by Equation (34),

$$\left\| \mathsf{D}\check{D}[x_0, u, t](\theta) \right\| \ =\ \left\| \mathsf{D}q_t(\theta) \right\| \ =\ \left\| \tilde{\mathsf{D}}G_t(\theta) \right\| \ \leq\ B_r\,.$$

Thus, for the constant B_r, every $u \in U_{\leq r}^{\omega}$ and $x_0 \in X_{\leq r}$ admit a full-measure subset $\hat{\Theta}_{r,u,x_0}$ on which every parameter-to-state function $\check{D}[x_0, u, t]$ is differentiable and has derivative norm bounded by B_r, uniformly over $t \in \mathbb{T}$.

Since $r \in \mathbb{R}^{\geq 0}$ was arbitrary, Definition 183 implies that D satisfies ae-uBIBSD. □

G.3.3 BIBLG: Graph-uniform selected-derivative bounds for loss gradients

Proposition 31 (Graph-uniform bounds imply selector-wise $\tilde{\mathsf{D}}$-BIBLG)**.** *Let S be a Euclidean parametric system, let $\mathcal{G}$ be an objective computation-graph family for S, let Γ be a graph-selector for $\mathcal{L}_S$ over $\mathcal{G}$, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator such that S satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG. Then,*

(1) the selected intensional derivative operator $\tilde{\mathsf{D}}^{\Gamma}$ is applicable to the family of objective functions $\mathcal{L}_S$;

(2) system S satisfies $\tilde{\mathsf{D}}^{\Gamma}$-BIBLG.

Proof. Since S satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG, Definition 195 implies, in particular, that $\tilde{\mathsf{D}}$ is intensionally-admissible for every computation graph in $\mathcal{G}$.

Let $q \in \mathcal{L}_S$ be arbitrary. Then, by definition of $\mathcal{L}_S$, there exist an admissible loss function $\ell \in \mathfrak{L}^S$ and a sample $z \in \mathcal{Z}^S$ such that

$$q = \mathcal{L}_z^\ell \,.$$

By the assumption on Γ, we have

$$\Gamma(q) = \Gamma\big(\mathcal{L}_z^\ell\big) \in \mathcal{G}\big[\mathcal{L}_{S,z}^\ell\big] \,.$$

Hence $\tilde{\mathsf{D}}$ is intensionally-admissible for $\Gamma(q)$.

Since $\Gamma(q)$ represents q, Proposition 24 implies that q is PAP, and also that

$$\tilde{\mathsf{D}}\big(\Gamma(q)\big) \in \partial_\bullet q \,.$$

By Definition 164, we have

$$\tilde{\mathsf{D}}^\Gamma q \;=\; \tilde{\mathsf{D}}\big(\Gamma(q)\big),$$

and hence

$$\tilde{\mathsf{D}}^\Gamma q \in \partial_\bullet q \,.$$

Since $q \in \mathcal{L}_S$ was arbitrary, $\tilde{\mathsf{D}}^\Gamma$ is applicable to $\mathcal{L}_S$ and it is a selected intensional derivative operator for this family.

It remains to establish the uniform bound. Let $\ell \in \mathfrak{L}^S$ and $r \in \mathbb{R}^{\geq 0}$ be arbitrary.

Since S satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG, there exists $B_{\ell,r} \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \big\|\tilde{\mathsf{D}}G(\theta)\big\| \leq B_{\ell,r} \qquad \forall\, z \in \mathcal{Z}_{\leq r}^S,\;\; \forall\, G \in \mathcal{G}\Big[\mathcal{L}_{S,z}^\ell\Big] \,.$$

In particular, by the assumption on Γ,

$$\Gamma\big(\mathcal{L}_{S,z}^\ell\big) \in \qquad \forall\, z \in \mathcal{Z}_{\leq r}^S,\; \forall\, \mathcal{G}\Big[\mathcal{L}_{S,z}^\ell\Big] \,.$$

Hence

$$\sup_{\theta \in \Theta_{\leq r}} \Big\|\tilde{\mathsf{D}}\Gamma\big(\mathcal{L}_z^\ell\big)(\theta)\Big\| \leq B_{\ell,r} \qquad \forall\, z \in \mathcal{Z}_{\leq r}^S \,.$$

By Definition 164, the above is equivalent to

$$\sup_{\theta \in \Theta_{\leq r}} \Big\|\tilde{\mathsf{D}}^\Gamma \mathcal{L}_z^\ell(\theta)\Big\| \leq B_{\ell,r} \qquad \forall\, z \in \mathcal{Z}_{\leq r}^S \,.$$

Since $\ell \in \mathfrak{L}^S$ and $r \in \mathbb{R}^{\geq 0}$ were arbitrary, Definition 193 implies that S satisfies $\tilde{\mathsf{D}}^\Gamma$-BIBLG. □

Proposition 32 (Graph-uniform loss-gradient bounds imply ae-BIBLG under a.e. correctness)**.** *Let S be a measurable-Euclidean parametric system, let $\mathcal{G}$ be an objective computation-graph family for S, and let $\tilde{\mathsf{D}}$ be a pseudo-derivative operator such that S satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG.*

Suppose that, for every $q \in \mathcal{L}_S$, every $G \in \mathcal{G}[q]$, and every $r \in \mathbb{R}^{\geq 0}$, there exists a full-measure subset

$$\hat{\Theta}_r^{q,G} \subseteq \Theta_{\leq r}$$

such that, for every $\theta \in \hat{\Theta}_r^{q,G}$,

$$q \text{ is differentiable at } \theta \qquad \text{and} \qquad \mathsf{D}q(\theta) = \tilde{\mathsf{D}}G(\theta) \,.$$

Then S satisfies ae-BIBLG.

Proof. Let $\ell \in \mathfrak{L}^S$ and $r \in \mathbb{R}^{\geq 0}$ be arbitrary. Since S satisfies $\mathcal{G}$-$\tilde{\mathsf{D}}$-BIBLG, there exists $B_{\ell,r} \in \mathbb{R}^{\geq 0}$ such that

$$\sup_{\theta \in \Theta_{\leq r}} \left\| \tilde{\mathsf{D}} G(\theta) \right\| \leq B_{\ell,r} \qquad \forall\, z \in \mathcal{Z}^S_{\leq r},\ \ \forall\, G \in \mathcal{G}\left[\mathcal{L}^\ell_{S,z}\right]. \tag{35}$$

Let $z \in \mathcal{Z}^S_{\leq r}$ be arbitrary, and let

$$q_z := \mathcal{L}^\ell_{S,z} \in \mathcal{L}_S.$$

Since $\mathcal{G}$ is an objective computation-graph family for S, there exists a computation graph

$$G_z \in \mathcal{G}[q_z]\,.$$

By the assumed a.e. correctness, there exists a full-measure subset

$$\hat{\Theta}^{q_z, G_z}_r \subseteq \Theta_{\leq r}$$

such that, for every $\theta \in \hat{\Theta}^{q_z, G_z}_r$,

$$q_z \text{ is differentiable at } \theta \qquad \text{and} \qquad \mathsf{D} q_z(\theta) = \tilde{\mathsf{D}} G_z(\theta)\,.$$

Hence, for every $\theta \in \hat{\Theta}^{q_z, G_z}_r$, Equation (35) gives

$$\left\| \mathsf{D} \mathcal{L}^\ell_{S,z}(\theta) \right\| = \left\| \mathsf{D} q_z(\theta) \right\| = \left\| \tilde{\mathsf{D}} G_z(\theta) \right\| \leq B_{\ell,r}\,.$$

Thus, for the constant $B_{\ell,r}$, every $z \in \mathcal{Z}^S_{\leq r}$ admits a full-measure subset of $\Theta_{\leq r}$ on which $\mathcal{L}^\ell_{S,z}$ is differentiable and its derivative norm is bounded by $B_{\ell,r}$.

Since $\ell \in \mathfrak{L}^S$ and $r \in \mathbb{R}^{\geq 0}$ were arbitrary, Definition 192 implies that S satisfies ae-BIBLG. □

H Additional Notions for MinMax RNCs

Table of Contents

H.1 MinMax Recurrence Operator 133
H.2 Parametric MinMax RNCs 133
H.3 Floating-point MinMax RNCs 134

H.1 MinMax Recurrence Operator

We introduce the following notation for the MinMax recurrence.

Definition 200 (MinMax recurrence operator)**.** The *MinMax recurrence operator* $\circledast$ is the binary operator

$$x \circledast (A, b) \;:=\; (A \otimes x) \oplus b$$

where $x \in \mathbb{R}^n$ and $(A, b) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$ for $n \in \mathbb{N}^{>0}$.

The *MinMax recurrence operator function* $\circledast_n$ of dimension $n \in \mathbb{N}^{>0}$ is the function

$$\circledast_n \,:\, \mathbb{R}^n \times (\mathbb{R}^{n\times n} \times \mathbb{R}^n) \to \mathbb{R}^n, \qquad\qquad \circledast_n(x, A, b) := x \circledast (A, b).$$

End of Definition 200.

H.2 Parametric MinMax RNCs

Definition 201 (Parametric MinMax recurrent units)**.** A *parametric MinMax recurrent unit* of state dimension n is Euclidean parametric dynamics

$$D = \langle X, U, f; \Theta \rangle$$

having Euclidean ambient spaces

$$X \subseteq \Omega_X = \mathbb{R}^n, \qquad U \subseteq \Omega_U = \mathbb{R}^m, \qquad \Theta \subseteq \Omega_\Theta = \mathbb{R}^p,$$

for some $m, p \in \mathbb{N}^{>0}$, together with continuous parametric functions

$$R : U \times \Theta \to \mathbb{R}^{n\times n}, \qquad\qquad s : U \times \Theta \to \mathbb{R}^n,$$

such that

$$f(x, u; \theta) = \circledast\big(x, R(u; \theta), s(u; \theta)\big) = (R(u; \theta) \otimes x) \oplus s(u; \theta)$$

for every $x \in X$, $u \in U$, and $\theta \in \Theta$.

We call R the *reset function* and s the *set function*, and refer to them jointly as the *input functions* of D, also writing

$$D = \langle X, U, f; \Theta \,|\, R, s \rangle$$

The unit D is a *parametric MinMax recurrent neuron* if its input functions (R, s) are analytic.

A MinMax recurrent unit is an instance $D[\theta]$ of D for $\theta \in \Theta$, and it is a *MinMax recurrent neuron* if its input functions $\big(R[\theta], s[\theta]\big)$ are analytic.

We write $\mathcal{U}^{\mathrm{p}}_{\mathrm{AN}}[\mathrm{MinMax}]$ for the class of parametric MinMax recurrent neurons, and we write $\mathcal{U}^{\mathrm{n}}_{\mathrm{MinMax}}$ for the class of MinMax recurrent neurons. *End of Definition 201.*

Definition 202 (Parametric MinMax recurrent neural cascades)**.** The class of *MinMax Recurrent Neural Cascade dynamics (MinMax RNC dynamics)* is

$$\mathrm{RNC}_{\mathrm{dyn}}\big(\mathcal{U}^{\mathrm{n}}_{\mathrm{MinMax}}\big),$$

and the class of *MinMax Recurrent Neural Cascades (MinMax RNCs)* is

$$\mathrm{RNC}\big(\mathcal{U}^{\mathrm{n}}_{\mathrm{MinMax}}\big).$$

The class of *parametric MinMax Recurrent Neural Cascade dynamics (parametric MinMax RNC dynamics)* is

$$\mathrm{RNC}_{\mathrm{dyn}}\big(\mathcal{U}^{\mathrm{p}}_{\mathrm{AN}}[\mathrm{MinMax}]\big),$$

and the class of *parametric MinMax Recurrent Neural Cascades (parametric MinMax RNCs)* is

$$\mathrm{RNC}\big(\mathcal{U}^{\mathrm{p}}_{\mathrm{AN}}[\mathrm{MinMax}]\big).$$

End of Definition 202.

Definition 203 (Parametric MinMax recurrent layers and cascades)**.** Let $\mathfrak{D}^{\mathrm{P}}_{\mathrm{MinMax}}$ denote the class of parametric MinMax recurrent units.

A *parametric MinMax recurrent layer* is a parametric recurrent layer whose units belong to $\mathfrak{D}^{\mathrm{P}}_{\mathrm{MinMax}}$, and a *parametric MinMax recurrent cascade* is a parametric recurrent cascade whose units belong to $\mathfrak{D}^{\mathrm{P}}_{\mathrm{MinMax}}$. *End of Definition 203.*

H.3 Floating-point MinMax RNCs

We first discuss the background, and then define floating-point MinMax RNCs.

H.3.1 Floating-point MLPs

We adopt all definitions from [60]. In particular, we adopt the definition of floating-point domains and operations from §2.1 and the definition of floating-point neural networks from §2.2. We refer to their floating-point neural networks as $\mathbb{F}$-MLPs. By their Corollaries 3.8 and 3.11, $\mathbb{F}$-MLPs can represent all functions over $\mathbb{F}^n$.

Note that $\mathbb{F}$-MLPs are not simply ordinary MLPs restricted to the floating-point subset of their real-valued domain. Rather, they are compositions of primitive functions supplied by the considered floating-point arithmetic.

H.3.2 Floating-point MinMax RNCs

We next introduce MinMax recurrent units and RNCs evaluated in floating-point arithmetic over a floating-point domain $\mathbb{F}$.

Definition 204 (Floating-point MinMax recurrent units)**.** Let $\mathbb{F}$ be a floating-point domain. A *floating-point MinMax recurrent unit over* $\mathbb{F}$ *of state dimension* n is dynamics

$$D = \langle X, U, f \rangle$$

having

$$X \subseteq \mathbb{F}^n, \qquad U \subseteq \mathbb{F}^m,$$

for some $m \in \mathbb{N}^{>0}$, together with input functions

$$R : U \to \mathbb{F}^{n \times n}, \qquad\qquad s : U \to \mathbb{F}^n,$$

such that

$$f(x, u) = \big(R(u) \otimes x\big) \oplus s(u)$$

for every $x \in X$ and $u \in U$.

We call R the *reset function* and s the *set function*, and refer to them jointly as the *input functions* of D, also writing

$$D = \langle X, U, f; R, s \rangle$$

The unit D is a *floating-point MinMax recurrent neuron* if R and s are $\mathbb{F}$-MLPs. *End of Definition 204.*

Definition 205 (Floating-point MinMax recurrent neural cascades)**.** Let $\mathbb{F}$ be a floating-point domain.

A *floating-point MinMax recurrent layer over* $\mathbb{F}$ is a recurrent layer whose units are floating-point MinMax recurrent units over $\mathbb{F}$. A *floating-point MinMax recurrent cascade over* $\mathbb{F}$ is a recurrent cascade whose units are floating-point MinMax recurrent units over $\mathbb{F}$.

A *floating-point MinMax Recurrent Neural Cascade (MinMax RNC) over* $\mathbb{F}$ is a floating-point MinMax recurrent cascade such that:

- every unit is a floating-point MinMax recurrent neuron over $\mathbb{F}$;
- every intermediate function and the final output function are $\mathbb{F}$-MLPs, under the canonical identification of their finite Cartesian product domains with subsets of some $\mathbb{F}^d$.

The *state degree* of a floating-point MinMax RNC is the maximum state dimension of any of its units. *End of Definition 205.*

We observe that the only functions introduced by a floating-point MinMax RNC in addition to those appearing in its $\mathbb{F}$-MLPs are the minimum and maximum operations. When evaluated on $\mathbb{F}$, these are native floating-point operations in the sense that they return one of their floating-point arguments and therefore introduce no additional rounding operation.

Part III

Theoretical Results for MinMax RNCs

I Expressivity of MinMax RNCs

In this appendix we provide full proofs of the results of Section 4.

- Section I.1 presents preliminary results.
  - Subsection I.1.1 develops the preliminary results for characterising the state degree of MinMax Neurons in terms of the permutation degree of a semiautomaton.
  - Subsections I.1.2, I.1.3 jointly provide preliminary results regarding the cascading dynamics while preserving the characteristics we are interested in to show results later.
- Section I.2 proves the theorems of Section 4.1, also making use of the preliminary results from Subsection I.1.1.
- Section I.3 focuses on realising casacades of semiautomata using cascades of MinMax Neurons, also making use of the preliminary results from Subsections I.1.2, I.1.3.
- Section I.4 proves the main expressivity theorem (Theorem 1), relying on all of the above.

Additionally, we prove that the expressivity results hold seamlessly for floating-point MinMax RNCs (Theorem 29 and Corollary 19).

I.1 Preliminary Results

I.1.1 Permutation Degree

We provide the more formally-precise version of Definition 3.

Definition 206 (See Def. 3)**.** The *minimal faithful permutation degree* of a group G, written $\mu(G)$, is the smallest number n such that there is an isomorphism between G and a subgroup of the symmetric group $\mathrm{Sym}([1..n])$. The *degree* of a permutation semiautomaton is the degree of its transition monoid (which is a group). The *degree* of a regular function F is the maximum between one and the degree of any simple group that is a factor of a subgroup of the transition monoid of the (semiautomaton of) the canonical automaton of F.

Theorem 21. *Let G be a group having minimal faithful permutation degree n, and let $A = \langle G, G, \delta\rangle$ be the semiautomaton with $\delta(x, a) = xa$. There exists $\varphi : G \to \mathrm{Sym}([1..n])$ such that A is realised by the semiautomaton $A' = \langle [1..n]^n, G, \delta'\rangle$ with $\delta'(\langle x_1, \ldots, x_n\rangle, g) = \langle x_{\pi(1)}, \ldots, x_{\pi(n)}\rangle$ for π the permutation $\pi = \varphi(g)$.*

Proof. For each $g \in G$, let

$$\delta_g : G \to G, \qquad \delta_g(x) = xg,$$

Then, the transition group of A is $G_A = \{\delta_g \mid g \in G\}$.

Since G has minimal faithful permutation degree equal to n, by Cayley's theorem (reported as Theorem 10 in Section A.15), there exists an isomorphism $\alpha : G \to H$ between G and a subgroup H of the symmetric group $\mathrm{Sym}([1..n])$.

Let us define the semiautomaton $A' = (H, G, \delta')$ where

$$\delta'(h, g) = h\alpha(g) \qquad (h \in H,\ g \in G).$$

Noting that the inverse function $\alpha^{-1} : H \to G$ is well-defined since α is bijective, we define the function $\zeta : H \times G \to G \times G$ as $\zeta(h, g) = \langle \alpha^{-1}(h), g\rangle$. Then we show that the tuple $(\alpha, \mathrm{id}, \zeta)$ is an assignment of $\mathcal{M}(A)$ into $\mathcal{M}(A')$.

Let $x, \sigma \in G$. It suffices to show

$$\text{(I)}\ \ \delta'(\alpha(x), \sigma) = \alpha(\delta(x, \sigma)),$$
$$\text{(II)}\ \ \zeta(\alpha(x), \sigma) = \langle x, \sigma\rangle.$$

We have that (I) holds since

$$\begin{aligned}\delta'(\alpha(x), \sigma) &= \alpha(x)\alpha(\sigma)\\ &= \alpha(x\sigma)\\ &= \alpha(\delta(x, \sigma)).\end{aligned}$$

We have that (II) holds since

$$\begin{aligned}\zeta(\alpha(x),\sigma) &= \langle \alpha^{-1}(\alpha(x)),\sigma\rangle \\ &= \langle x,\sigma\rangle.\end{aligned}$$

Now, let us define the semiautomaton $A'' = ([1..n]^n, G, \delta'')$ where

$$\delta''(\langle i_1,\dots,i_n\rangle, g) := \langle i_{\pi(1)},\dots,i_{\pi(n)}\rangle \quad \text{for } \pi = \alpha(g).$$

We define the function $\alpha' : H \to [1..n]^n$ as $\alpha'(h) = \langle h(1),\dots,h(n)\rangle$, and we define the function $\zeta' : [1..n]^n \times G \to G \times G$ as $\zeta'(\langle i_1,\dots,i_n\rangle, g) := \langle \pi, g\rangle$ for $\pi : [1..n] \to [1..n]$ such that $\pi(j) = i_j$. Then we show that the tuple $(\alpha', \mathrm{id}, \zeta')$ is an assignment of $\mathcal{M}(A')$ into $\mathcal{M}(A'')$.

Let $h \in H, g \in G$. It suffices to show

$$\begin{aligned}&\text{(I)}\ \ \delta''(\alpha'(h),g) = \alpha'(\delta'(h,g)),\\ &\text{(II)}\ \ \zeta'(\alpha'(g),g) = \langle h,g\rangle.\end{aligned}$$

We have that (I) holds since

$$\begin{aligned}\delta''(\alpha'(h),g) &= \delta''(\langle h(1),\dots,h(n)\rangle, g)\\ &= \langle h(\pi(1)),\dots,h(\pi(n))\rangle \quad \text{for } \pi = \alpha(g)\\ &= \langle (h\pi)(1),\dots,(h\pi)(n)\rangle\\ &= \alpha'(h\pi)\\ &= \alpha'(h\alpha(g))\\ &\alpha'\big(\delta'(h,g)\big).\end{aligned}$$

We have that (II) holds since

$$\begin{aligned}\zeta'(\alpha'(h),g) &= \zeta'(\langle h(1),\dots,h(n)\rangle, g)\\ &= \zeta'(\pi, g)\end{aligned}$$

for π such that $\pi(i) = h(i)$ for all i, but then $\pi = h$, and hence $\zeta'(\alpha'(h),g) = \zeta'(h,g)$ as required.

Overall, we have shown that A'' realises A', that in turn realises A. Thus, by transitivity, we conclude that A'' realises A. Furthermore, we have that $\varphi := \alpha' \circ \alpha$ is a function $G \to \mathrm{Sym}([1..n])$. Thus, φ and A'' witness the claim of the theorem. □

Theorem 22. *Let $A = \langle Q, \Sigma, \delta\rangle$ be a permutation semiautomaton of degree n. There exists $\varphi : \Sigma \to \mathrm{Sym}([1..n])$ such that A is realised by the semiautomaton $A' = \langle [1..n]^n, \Sigma, \delta'\rangle$ with $\delta'(\langle x_1,\dots,x_n\rangle, g) = \langle x_{\pi(1)},\dots,x_{\pi(n)}\rangle$ for π the permutation $\pi = \varphi(\sigma)$.*

Proof. The transition monoid $\mathsf{M}(A) = (G, \circ)$ of A is a group of degree n. By Theorem 21 there is $\varphi' : G \to \mathrm{Sym}([1..n])$ such that the $\mathsf{M}(A)$-semiautomaton is realised by the semiautomaton $A' = \langle [1..n]^n, G, \delta'\rangle$ with $\delta'(\langle x_1,\dots,x_n\rangle, g) = \langle x_{\pi(1)},\dots,x_{\pi(n)}\rangle$ for π the permutation $\pi' = \varphi(g)$. Then A' realises A since it realises the $\mathsf{M}(A)$-semiautomaton, which realises A since it has a superset of its transformations. Finally, $\varphi := \psi \circ \varphi'$ where $\psi(\sigma) := \delta_\sigma \in G$. □

I.1.2 Composition with an input function

Definition 207. Dynamics D and D' are *strongly equivalent* if $\mathcal{C}_\eta(D) = \mathcal{C}_\eta(D')$ (which also implies $X = X'$ and $U = U'$).

Definition 208. Given dynamics $D = \langle X, U, f\rangle$ and a function $\varphi : V \to U$, the composition $\varphi \circ D$ yields the dynamics $D_\varphi = \langle X, V, f_\varphi\rangle$ with $f_\varphi(x, v) = f(x, \varphi(v))$.

Definition 209. A class of functions Ψ satifies the *universal approximation property* if, for every continuous function $f : X \subseteq \mathbb{R}^m \to \mathbb{R}^n$ with X compact and for every $\epsilon > 0$, class Ψ contains a function $\hat{f} : X \to \mathbb{R}^n$ such that $\|f(x) - \hat{f}(x)\| \leq \epsilon$ for every $x \in X$.

Proposition 33. *Let Ψ be a class of functions satisfying the universal approximation property, let D be η-finite dynamics with input space U, let U' be an η-finite space, and let $\varphi : U' \to U$ be a continuous function. Furthermore, suppose that each η-component U^η of U satisfies $\bar{B}_{\Omega_U}(u) \subseteq U^\eta$ for some $u \in U$. Then, the dynamics $\varphi \circ D$ admit strongly equivalent dynamics $\psi \circ D$ for some $\psi \in \Psi$.*

Proof. Let $D = \langle X, U, f\rangle$. Then, naming $D_\varphi := \varphi \circ D$, we have $D_\varphi = \langle X, U', f_\varphi\rangle$ with $f_\varphi(x, u') = f(x, \varphi(u'))$

Let $U^\eta_1, \dots, U^\eta_n$ be the η-components of U. By assumption, there exist $u_1, \dots, u_n$ such that $B(u_i, \epsilon) \subseteq U^\eta_i$ for every $i \in [1..n]$. Let $\varphi' : U' \to \{u_1, \dots, u_n\}$ be the function defined as $\varphi'(u') = u_i$ for $i \in [1..n]$ such that $\varphi(u') \in U^\eta_i$. Fitst, we have that φ' is well-defined since U is partitioned by its η-components. Second, we have $\overline{\varphi'} = \overline{\varphi}$ since $\overline{\varphi'}(\overline{u'}) = \overline{\varphi'(u')}$ and $\varphi'(u') \in \overline{\varphi(u')}$ by construction. Third, we have that φ' is continuous; in fact, by continuity of φ, we have that $u'_1 \sim_{U'} u'_2$ implies $\varphi(u'_1) \sim_U \varphi(u'_2)$, hence φ' is constant on the η-components of U', and hence φ' is continuous by Lemma 22 of [37].

Then, since we have shown that φ' is continuous, there exists $\psi \in \Psi$ that is an ϵ-approximation of φ'.

Letting $D_\psi := \psi \circ D$, we have $D_\psi = \langle X, U', f_\psi\rangle$ with $f_\psi(x, u') = f(x, \psi(u'))$.

Since D_φ and D_ψ have the same state and input space, it suffices to show $\overline{f_\varphi} = \overline{f_\psi}$.

First we show $\overline{\psi} = \overline{\varphi}$. Namely, letting $u' \in U'$, we show that $\overline{\psi}(\overline{u'}) = \overline{\varphi}(\overline{u'})$. First, by the approximation assumption, we have

$$\psi(u') \in \bar{B}_{\Omega_U}\big(\varphi'(u'), \epsilon\big) \subseteq \overline{\varphi'(u')} = \overline{\varphi'}(\overline{u'}) = \overline{\varphi}(\overline{u'}),$$

with the last equality argued above. Hence $\psi(u') \in \overline{\varphi}(\overline{u'})$, which implies $\overline{\psi(u')} = \overline{\varphi}(\overline{u'})$. Then,

$$\overline{\psi}(\overline{u'}) = \overline{\psi(u')} = \overline{\varphi}(\overline{u'}),$$

which concludes the proof of the claim $\overline{\psi} = \overline{\varphi}$.

Finally we show $\overline{f_\varphi} = \overline{f_\psi}$ as required. By the two following chains of equalities,

$$\overline{f_\varphi}(\overline{x}, \overline{u'}) = \overline{f_\varphi(x, u')} = \overline{f(x, \varphi(u'))} = \overline{f}(\overline{x}, \overline{\varphi(u')}),$$
$$\overline{f_\psi}(\overline{x}, \overline{u'}) = \overline{f_\psi(x, u')} = \overline{f(x, \psi(u'))} = \overline{f}(\overline{x}, \overline{\psi(u')}),$$

we conclude $\overline{f_\varphi} = \overline{f_\psi}$ since we have shown above $\overline{\varphi} = \overline{\psi}$. □

Remark 24. Under strong equivalence, we can replace D with D' anywhere, e.g., in a cascade.

I.1.3 Cascades

Lemma 15. *Let us consider a parallel composition $D = (D_1 \parallel \cdots \parallel D_n)$ where each D_i is η-finite. Let U be the input space of D, and let m be the number of η-components of U. For every $i \in [1..n]$, let D'_i be dynamics that can realise D_i. Then, for every η-finite space $\hat{U}$ whose number of η-components is m, there exist continuous functions $\phi_1, \dots, \phi_n$ with domain $\hat{U}$ such that D can be realised by the composition $D' = (\phi_1 \circ D'_1 \parallel \cdots \parallel \phi_n \circ D'_n)$.*

Proof. Let us note that, for each $i \in [1..n]$, the dynamics D_i are of the form $D_i = \langle X_i, U, f_i\rangle$. In particular, all dynamics $D_1, \dots, D_n$ share the same input space as they are part of a parallel composition. For each $i \in [1..n]$, let dynamics D'_i be of the form $D'_i = \langle X'_i, U'_i, f'_i\rangle$. Then, for each $i \in [1..n]$, since D'_i is a realisation of D_i, there exists an assignment $(\alpha_i, \iota_i, \zeta_i)$ of $\mathcal{M}(\mathcal{C}_\eta(D_i)) = \langle \overline{X}_i, \overline{U}, \overline{f}_i, \overline{X}_i \times \overline{U}_i, \mathrm{id}\rangle$ into $\mathcal{M}(\mathcal{C}_\eta(D'_i)) = \langle \overline{X}'_i, \overline{U}'_i, \overline{f}'_i, \overline{X}'_i \times \overline{U}'_i, \mathrm{id}\rangle$.

Let $U^\eta_1, \dots, U^\eta_m$ be the η-components of U, let $\hat{U}$ be an η-finite space whose number of η-components is m, and let $\hat{U}^\eta_1, \dots, \hat{U}^\eta_m$ be its η-components. Let $u_1, \dots, u_m$ be such that $u_i \in U^\eta_i$ for every $i \in [1..m]$, and hence $\overline{u_i} = U^\eta_i$. Similarly, let $\hat{u}_1, \dots, \hat{u}_m$ be such that $\hat{u}_i \in \hat{U}^\eta_i$ for every $i \in [1..m]$, and hence $\overline{\hat{u}_i} = \hat{U}^\eta_i$.

Then let us define $\iota : \overline{U} \to \overline{\hat{U}}$ as $\iota(\overline{u_j}) = \overline{\hat{u}_j}$, to be used below, also noting that it is a bijection and hence it admits an inverse $\iota^{-1} : \overline{\hat{U}} \to \overline{U}$.

For every $i \in [1..n]$, noting that ι_i is of the form $\iota_i : \overline{U} \to \overline{U'_i}$, and letting $u'_{i,1}, \ldots, u'_{i,m_i} \in U'_i$ be arbitrarily-chosen elements such that $\overline{u'_{i,1}}, \ldots, \overline{u'_{i,m_i}}$ are the η-components of U'_i, we define $\phi_i : \hat{U} \to U_i$ as $\phi_i(\hat{u}) := u'_{i,k}$ such that $u'_{i,k} \in \iota_i\big(\iota^{-1}(\overline{\hat{u}})\big)$. We have that each ϕ_i is a well-defined continuous function. First, we have that each ϕ_i is well-defined since there exists a unique $k \in [1..m_i]$ such that $u'_{i,k} \in \iota_i\big(\iota^{-1}(\overline{\hat{u}})\big)$ by our choice of $u'_{i,1}, \ldots, u'_{i,m_i}$. Second, we have that each ϕ_i is continuous by Lemma 22 of [37].

Then, we have that $\overline{\phi_i} = \iota_i \circ \iota^{-1}$; in fact, we have $\overline{\phi_i}(\overline{\hat{u}}) = \iota_i(\iota^{-1}(\overline{\hat{u}}))$ since $\phi_i(\hat{u}) = u'_{i,k}$ with $u'_{i,k} \in \iota_i\big(\iota^{-1}(\overline{\hat{u}})\big)$ by definition, and hence $\overline{\phi_i}(\overline{\hat{u}}) = \overline{u'_{i,k}} = \iota_i\big(\iota^{-1}(\overline{\hat{u}})\big)$.

Let us note that D and D' are of the following form

$$D = \langle X, U, f\rangle \quad \text{with} \quad X = X_1 \times \cdots \times X_n$$
$$D' = \langle X', U, f'\rangle \quad \text{with} \quad X' = X'_1 \times \cdots \times X'_n.$$

We show that D' realises D, by showing an assignment (α, ι, ζ) of the η-canonical semiautomaton $\mathcal{C}_\eta(D)$ into the η-canonical semiautomaton $\mathcal{C}_\eta(D')$. Let us note that $\mathcal{C}_\eta(D)$ and $\mathcal{C}_\eta(D')$ are of the following form

$$\mathcal{C}_\eta(D) = \langle \overline{X}, \overline{U}, \overline{f}\rangle \quad \text{with} \quad \overline{X} = \overline{X}_1 \times \cdots \times \overline{X}_n,$$
$$\mathcal{C}_\eta(D') = \langle \overline{X}', \overline{U}, \overline{f}'\rangle \quad \text{with} \quad \overline{X}' = \overline{X}'_1 \times \cdots \times \overline{X}'_n.$$

We have defined above the function ι, and next we define the two remaining functions for an assignment (α, ι, ζ),

$$\alpha : \overline{X} \to \mathcal{P}_+\big(\overline{X}'\big) \qquad \text{as } \alpha := \alpha_1 \times \cdots \times \alpha_n,$$
$$\zeta : \big(\overline{X}' \times \overline{\hat{U}}\big) \to \big(\overline{X} \times \overline{U}\big) \qquad \text{as } \zeta\big(\langle \overline{x}'_1, \ldots, \overline{x}'_n\rangle, \overline{\hat{u}}\big) := \big\langle \langle \overline{x}_1, \ldots, \overline{x}_n\rangle, \iota^{-1}(\overline{\hat{u}})\big\rangle$$
$$\text{for } \langle \overline{x}_i, _\rangle = \zeta_i(\overline{x}'_i, \overline{\phi_i}(\hat{u})).$$

Let us consider

$$\langle \overline{x}_1, \ldots, \overline{x}_n\rangle \in \overline{X}, \qquad \overline{u} \in \overline{U}, \qquad \langle \overline{x}'_1, \ldots, \overline{x}'_n\rangle \in \alpha\big(\langle \overline{x}_1, \ldots, \overline{x}_n\rangle\big),$$
$$\big\langle \overline{x}_{1,\mathrm{new}}, \ldots, \overline{x}_{m,\mathrm{new}}\big\rangle = \overline{f}\big(\langle \overline{x}_1, \ldots, \overline{x}_n\rangle, \overline{u}\big), \qquad \big\langle \overline{x}'_{1,\mathrm{new}}, \ldots, \overline{x}'_{n,\mathrm{new}}\big\rangle = \overline{f}'\big(\langle \overline{x}'_1, \ldots, \overline{x}'_n\rangle, \overline{u}\big).$$

By Property (I) of assignment, for every $i \in [1..n]$,

$$\overline{x}'_{i,\mathrm{new}} = \overline{f}'_i\big(\overline{x}'_i, \iota_i(\overline{u})\big) \in \alpha_i\big(\overline{f}_i(\overline{x}_i, \overline{u})\big),$$

and hence

$$\begin{aligned}
\big\langle \overline{x}'_{1,\mathrm{new}}, \ldots, \overline{x}'_{n,\mathrm{new}}\big\rangle &= \big\langle \overline{f}'_1\big(\overline{x}'_1, \overline{\phi_1}(\iota(\overline{u}))\big), \ldots, \overline{f}'_n\big(\overline{x}'_n, \overline{\phi_n}(\iota(\overline{u}))\big)\big\rangle \\
&= \big\langle \overline{f}'_1\big(\overline{x}'_1, \iota_1(\iota^{-1}(\iota(\overline{u})))\big), \ldots, \overline{f}'_n\big(\overline{x}'_n, \iota_n(\iota^{-1}(\iota(\overline{u})))\big)\big\rangle \quad \text{| Shown above } \overline{\phi_i} = \iota_i \circ \iota^{-1} \\
&= \big\langle \overline{f}'_1\big(\overline{x}'_1, \iota_1(\overline{u})\big), \ldots, \overline{f}'_n\big(\overline{x}'_n, \iota_n(\overline{u})\big)\big\rangle \\
&\in \Big(\alpha_1\big(\overline{f}'_1\big(\overline{x}'_1, \iota_1(\overline{u})\big)\big) \times \cdots \times \alpha_n\big(\overline{f}'_n\big(\overline{x}'_n, \iota_n(\overline{u})\big)\big)\Big) \\
&= \alpha\big(\overline{f}\big(\langle \overline{x}_1, \ldots, \overline{x}_n\rangle, \overline{u}\big)\big),
\end{aligned}$$

showing that the constructed assignment (α, ι, ζ) satisfies Property (I) as an assignment of $\mathcal{C}(D)$ into $\mathcal{C}(D')$.

For every $i \in [1..n]$, by definition of α, we have that $\langle \overline{x}'_1, \ldots, \overline{x}'_n \rangle \in \alpha\big(\langle \overline{x}_1, \ldots, \overline{x}_n \rangle\big)$ implies $\overline{x}'_i \in \alpha_i(\overline{x}_i)$, and hence by Property (II) of assignment,

$$\zeta_i(\overline{x}'_i, \iota_i(\overline{u})) = \langle \overline{x}_i, \overline{u} \rangle.$$

Then, by definition of ζ,

$$\zeta\big(\langle \overline{x}'_1, \ldots, \overline{x}'_n \rangle, \iota(\overline{u})\big) = \big\langle \langle \overline{x''_1}, \ldots, \overline{x''_n} \rangle, \iota^{-1}(\iota(\overline{u})) \big\rangle \quad \text{for} \quad \langle \overline{x''_i}, _ \rangle = \zeta_i(\overline{x}'_i, \overline{\phi_i}(\iota(\overline{u}))).$$

We have $\overline{x''}_i = \overline{x}_i$ since $\overline{\phi_i} = \iota_i \circ \iota^{-1}$, and hence $\zeta_i(\overline{x}'_i, \overline{\phi_i}(\iota(\overline{u}))) = \zeta_i(\overline{x}'_i, \iota_i(\overline{u})) = \langle \overline{x}_i, \overline{u} \rangle$. Also, $\iota^{-1}(\iota(\overline{u})) = \overline{u}$. Therefore, the equation above becomes

$$\zeta\big(\langle \overline{x}'_1, \ldots, \overline{x}'_n \rangle, \iota(\overline{u})\big) = \big\langle \langle \overline{x_1}, \ldots, \overline{x_n} \rangle, \overline{u} \big\rangle,$$

as required to show that the constructed assignment (α, ι, ζ) satisfies Property (II) as an assignment of $\mathcal{C}(D)$ into $\mathcal{C}(D')$. Therefore (α, ι, ζ) is an assignment of $\mathcal{C}(D)$ into $\mathcal{C}(D')$ as required. □

Lemma 16. *Let us consider a cascade*

$$C = \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big),$$

where each $D_{i,j}$ is η-finite. Let U be the input space of D, and let m be the number of η-components of U. For every $i \in [1..d]$ and every $j \in [1..m_i]$, let $D'_{i,j}$ be dynamics that can realise $D_{i,j}$. Then, for every η-finite space $\hat{U}$ whose number of η-components is m, there exist continuous functions $\phi_{i,j}$ such that C can be realised by

$$C' = \big(\phi_{1,1} \circ D'_{1,1} \parallel \cdots \parallel \phi_{1,m_1} \circ D'_{1,m_1}\big) \ltimes \cdots \ltimes \big(\phi_{d,1} \circ D'_{d,1} \parallel \cdots \parallel \phi_{d,m_d} \circ D'_{d,m_d}\big)$$

and every $\phi_{1,j}$ has domain $\hat{U}$.

Proof. For each $i \in [1..d]$, let us name

$$C_i := \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{i,1} \parallel \cdots \parallel D_{i,m_{d-1}}\big),$$

For each $i \in [1..d]$ and each $j \in [1..m_i]$, let $D_{i,j} = \langle X_{i,j}, U_i, f_{i,j} \rangle$, noting that dynamics occurring in the same parallel composition share the same input space.

We show the claim by induction on d. In the base case $d = 1$ and the claim holds by Lemma 15. In the inductive case $d > 1$ and we assume the claim holds for $d - 1$.

By Lemma 15, there exist continuous functions $\varphi_{d,1}, \ldots, \varphi_{d,m_d}$ with domain U_{d-1} such that

$$\hat{D}_d := \big(\varphi_{d,1} \circ D'_{d,1} \parallel \cdots \parallel \varphi_{d,m_d} \circ D'_{d,m_d}\big)$$

can realise

$$D_d := \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big).$$

Since $\hat{D}_d$ can realise D_d, there exists an assignment $(\alpha_d, \iota_d, \zeta_d)$ of $\mathcal{M}(\mathcal{C}_\eta(D_d))$ into $\mathcal{M}(\mathcal{C}_\eta(\hat{D}_d))$.

Furthermore, by the inductive hypothesis, there exist functions $\phi_{i,j}$ such that the cascade

$$C'_{d-1} = \big(\phi_{1,1} \circ D'_{1,1} \parallel \cdots \parallel \phi_{1,m_1} \circ D'_{1,m_1}\big) \ltimes \cdots \ltimes \big(\phi_{d-1,1} \circ D'_{d-1,1} \parallel \cdots \parallel \phi_{d-1,m_{d-1}} \circ D'_{d-1,m_d}\big)$$

can realise C_{d-1}. In particular, we can choose each function $\phi_{1,j}$ to have domain $\hat{U}$, and each function $\phi_{i,j}$ for $i > 1$ to have domain U_i. Since C'_{d-1} can realise C_{d-1}, there exists an assignment $(\alpha_{\text{ih}}, \iota_{\text{ih}}, \zeta_{\text{ih}})$ of $\mathcal{M}(\mathcal{C}_\eta(C_{d-1}))$ into $\mathcal{M}(\mathcal{C}_\eta(C'_{d-1}))$,

Let us choose a continuous function $\varphi : U_d \to U_d$ and $\psi : \mathbf{X}'_{d-1} \times \hat{U} \to U_d$ such that $\overline{\varphi} = \iota_d$ and $\overline{\psi} = \zeta_{\text{ih}}$, which exist by Lemma 22 of [37]. Then, for each $i \in [1..m_d]$, let us define $\phi_{d,i} := \varphi_{d,i} \circ \varphi \circ \psi$. Then, let

$$D'_d := \big(\phi_{d,1} \circ D'_{d,1} \parallel \cdots \parallel \phi_{d,m_d} \circ D'_{d,m_d}\big),$$

and let us define

$$C'_{d-1} := \big(D'_{1,1} \parallel \cdots \parallel D'_{1,m_1}\big) \ltimes \cdots \ltimes \big(D'_{d-1,1} \parallel \cdots \parallel D'_{d-1,m_{d-1}}\big),$$
$$C' := C'_{d-1} \ltimes D'_d.$$

Also, let us note that

$$C = C_{d-1} \ltimes D_d.$$

The above dynamics are of the following form,

$$\begin{aligned}
C_{d-1} &= \langle \mathbf{X}_{d-1}, U, \mathbf{f}_{d-1} \rangle, \\
C'_{d-1} &= \langle \mathbf{X}'_{d-1}, \hat{U}, \mathbf{f}'_{d-1} \rangle, \\
D_d &= \langle X_d, U_d, f_d \rangle \quad \text{with} \quad U_d = U \times \mathbf{X}_{d-1}, \\
D'_d &= \langle X'_d, U'_d, f'_d \rangle \quad \text{with} \quad U'_d = \hat{U} \times \mathbf{X}'_{d-1},
\end{aligned}$$

and hence

$$\begin{aligned}
C_d &= \langle \mathbf{X}_{d-1} \times X_d,\ U,\ \mathbf{f}_{d-1} \ltimes f_d \rangle, \\
C'_d &= \langle \mathbf{X}'_{d-1} \times X'_d,\ \hat{U},\ \mathbf{f}'_{d-1} \ltimes f'_d \rangle.
\end{aligned}$$

Finally, let us note

$$\begin{aligned}
\mathcal{M}(\mathcal{C}_\eta(D_d)) &= \big\langle \overline{X_d},\ \overline{U_d},\ \overline{f_d},\ \overline{X_d} \times \overline{U_d},\ \mathrm{id} \big\rangle, \\
\mathcal{M}(\mathcal{C}_\eta(D'_d)) &= \big\langle \overline{X'_d},\ \overline{U'_d},\ \overline{f'_d},\ \overline{X'_d} \times \overline{U'_d},\ \mathrm{id} \big\rangle, \\
\mathcal{M}(\mathcal{C}_\eta(C_{d-1})) &= \big\langle \overline{\mathbf{X}_{d-1}} \times \overline{X_d},\ \overline{U},\ \overline{\mathbf{f}_{d-1}} \ltimes \overline{f_d},\ \overline{\mathbf{X}_{d-1}} \times \overline{X_d} \times \overline{U},\ \mathrm{id} \big\rangle, \\
\mathcal{M}(\mathcal{C}_\eta(C'_{d-1})) &= \big\langle \overline{\mathbf{X}'_{d-1}} \times \overline{X'_d},\ \overline{\hat{U}},\ \overline{\mathbf{f}'_{d-1}} \ltimes \overline{f'_d},\ \overline{\mathbf{X}'_{d-1}} \times \overline{X'_d} \times \overline{\hat{U}},\ \mathrm{id} \big\rangle.
\end{aligned}$$

We show that C' can realise C by showing that the triple (α, ι, ζ) defined below is an assignment of $\mathcal{M}(\mathcal{C}_\eta(C_d))$ into $\mathcal{M}(\mathcal{C}_\eta(C'_d))$,

$$\begin{aligned}
&\alpha : \overline{\mathbf{X}_{d-1}} \times \overline{X_d} \to \mathcal{P}_+\big(\overline{\mathbf{X}'_{d-1}} \times \overline{X'_d}\big) && \text{as } \alpha := \alpha_{\mathrm{ih}} \times \alpha_d, \\
&\iota : \overline{U} \to \overline{\hat{U}} && \text{as } \iota := \iota_{\mathrm{ih}}, \\
&\zeta : \big(\overline{\mathbf{X}'_{d-1}} \times \overline{X'_d} \times \overline{\hat{U}}\big) \to \big(\overline{\mathbf{X}_{d-1}} \times \overline{X_d} \times \overline{U}\big) && \text{as } \zeta\big(\langle \overline{\mathbf{x}'_{d-1}}, \overline{x'_d} \rangle, \overline{\hat{u}}\big) := \langle c, b, a \rangle \\
& && \text{for } \langle a, b, c \rangle = \zeta_d\big(\overline{x'_d}, \iota_d(\zeta_{\mathrm{ih}}(\overline{\mathbf{x}'_{d-1}}, \overline{\hat{u}}))\big).
\end{aligned}$$

Let us consider

$$\overline{u},\ \ \overline{\hat{u}},\ \ \overline{\mathbf{x}_{d-1}},\ \ \overline{x_d},\ \ \overline{\mathbf{x}'_{d-1}},\ \ \overline{x'_d}$$

such that

$$\begin{aligned}
&\overline{u} \in \overline{U}, \\
&\overline{\hat{u}} = \iota(\overline{u}) && \text{i.e., } \overline{\hat{u}} = \iota_{\mathrm{ih}}(\overline{u}), \\
&\langle \overline{\mathbf{x}_{d-1}}, \overline{x_d} \rangle \in \overline{\mathbf{X}_{d-1}} \times \overline{X_d}, \\
&\langle \overline{\mathbf{x}'_{d-1}}, \overline{x'_d} \rangle \in \alpha\big(\langle \overline{\mathbf{x}_{d-1}}, \overline{x_d} \rangle\big), && \text{i.e., } \overline{\mathbf{x}'_{d-1}} \in \alpha_{\mathrm{ih}}\big(\overline{\mathbf{x}_{d-1}}\big) \ \wedge\ \overline{x'_d} \in \alpha_d\big(\overline{x_d}\big).
\end{aligned}$$

Furthermore, let us consider

$$\overline{\mathbf{x}_{d-1,\mathrm{new}}},\ \ \overline{x_{d,\mathrm{new}}},\ \ \overline{\mathbf{x}'_{d-1,\mathrm{new}}},\ \ \overline{x'_{d,\mathrm{new}}}$$

such that

$$\big\langle \overline{\mathbf{x}_{d-1,\mathrm{new}}}, \overline{x_{d,\mathrm{new}}} \big\rangle = \big\langle \overline{\mathbf{f}_{d-1}}\big(\overline{\mathbf{x}_{d-1}}, \overline{u}\big),\ \overline{f_d}\big(\overline{x_d}, \langle \overline{\mathbf{x}_{d-1}}, \overline{u} \rangle\big) \big\rangle, \tag{36}$$

$$\big\langle \overline{\mathbf{x}'_{d-1,\mathrm{new}}}, \overline{x'_{d,\mathrm{new}}} \big\rangle = \big\langle \overline{\mathbf{f}'_{d-1}}\big(\overline{\mathbf{x}'_{d-1}}, \overline{\hat{u}}\big),\ \overline{f'_d}\big(\overline{x'_d}, \langle \overline{\mathbf{x}'_{d-1}}, \overline{\hat{u}} \rangle\big) \big\rangle. \tag{37}$$

To show Property (I) of assignment, it suffices to show

$$\begin{aligned}
&\big\langle \overline{\mathbf{x}'_{d-1,\mathrm{new}}}, \overline{x'_{d,\mathrm{new}}} \big\rangle \in \alpha\big(\langle \overline{\mathbf{x}_{d-1,\mathrm{new}}}, \overline{x_{d,\mathrm{new}}} \rangle\big), \\
&\text{i.e., } \overline{\mathbf{x}'_{d-1,\mathrm{new}}} \in \alpha_{\mathrm{ih}}\big(\overline{\mathbf{x}_{d-1,\mathrm{new}}}\big) \ \wedge\ \overline{x'_{d,\mathrm{new}}} \in \alpha_d\big(\overline{x_{d,\mathrm{new}}}\big).
\end{aligned}$$

First, we have

$$\begin{aligned}
\overline{\mathbf{x}'_{d-1,\mathrm{new}}} &= \overline{\mathbf{f}'_{d-1}}\big(\overline{\mathbf{x}'_{d-1}}, \hat{\overline{u}}\big) && \text{| (37)}\\
&= \overline{\mathbf{f}'_{d-1}}\big(\overline{\mathbf{x}'_{d-1}}, \overline{\iota_{\mathrm{ih}}}(\overline{u})\big) && \text{| def. of } \iota\\
&\in \alpha_{\mathrm{ih}}\big(\overline{\mathbf{f}_{d-1}}\big(\overline{\mathbf{x}_{d-1}}, \overline{u}\big)\big). && \text{| property (I) since } \overline{\mathbf{x}'_{d-1}} \in \alpha_{\mathrm{ih}}(\overline{\mathbf{x}_{d-1}})\\
&= \alpha_{\mathrm{ih}}\big(\overline{\mathbf{x}_{d-1,\mathrm{new}}}\big). && \text{| (36)}
\end{aligned}$$

Then, we have

$$\begin{aligned}
\overline{x'_{d,\mathrm{new}}} &= \overline{f'_d}\big(\overline{x'_d}, \langle \overline{\mathbf{x}'_{d-1}}, \hat{\overline{u}}\rangle\big) && \text{| (37)}\\
&= \overline{f'_d}\big(\overline{x'_d}, \langle \overline{\mathbf{x}'_{d-1}}, \iota(\overline{u})\rangle\big) && \text{| def. of } \hat{\overline{u}}\\
&= \overline{\hat{f}_d}\big(\overline{x'_d}, \overline{\varphi}(\overline{\psi}(\overline{\mathbf{x}'_{d-1}}, \iota(\overline{u})))\big) && \text{| def. of } f'_d\text{, letting } \hat{f}_d \text{ the dynamics func. of } \hat{D}_d\\
&= \overline{\hat{f}_d}\big(\overline{x'_d}, \iota_d(\zeta_{\mathrm{ih}}(\overline{\mathbf{x}'_{d-1}}, \iota(\overline{u})))\big) && \text{| def. of } \varphi \text{ and } \psi\\
&= \overline{\hat{f}_d}\big(\overline{x'_d}, \iota_d(\zeta_{\mathrm{ih}}(\overline{\mathbf{x}'_{d-1}}, \iota_{\mathrm{ih}}(\overline{u})))\big) && \text{| def. of } \iota\\
&= \overline{\hat{f}_d}\big(\overline{x'_d}, \iota_d(\overline{\mathbf{x}_{d-1}}, \overline{u})\big) && \text{| property (II) since } \overline{\mathbf{x}'_{d-1}} \in \alpha_{\mathrm{ih}}(\mathbf{x}_{d-1})\\
&\in \alpha_d\big(\overline{f_d}\big(\overline{x_d}, \langle \overline{\mathbf{x}_{d-1}}, \overline{u}\rangle\big)\big) && \text{| property (I) since } \overline{x'_d} \in \alpha_d(\overline{x_d})\\
&= \alpha_d\big(\overline{x_{d,\mathrm{new}}}\big). && \text{| (36)}
\end{aligned}$$

Thus, Property (I) holds for the constructed assignment (α, ι, ζ).

Next we show Property (II). By definition of ζ, we have

$$\begin{aligned}
&\zeta\big(\langle \overline{\mathbf{x}'_{d-1}}, \overline{x'_d}\rangle, \hat{\overline{u}}\big) = \langle c, b, a\rangle,\\
&\text{where } \langle a, b, c\rangle = \zeta_d\big(\overline{x'_d},\ \iota_d(\zeta_{\mathrm{ih}}(\overline{\mathbf{x}'_{d-1}}, \hat{\overline{u}}))\big).
\end{aligned}$$

Then,

$$\begin{aligned}
\zeta_{\mathrm{ih}}(\overline{\mathbf{x}'_{d-1}}, \hat{\overline{u}}) &= \zeta_{\mathrm{ih}}(\overline{\mathbf{x}'_{d-1}}, \iota_{\mathrm{ih}}(\overline{u})) && \text{| def. of } \hat{\overline{u}}\\
&= \langle \overline{\mathbf{x}_{d-1}}, \overline{u}\rangle, && \text{| Property (II)}
\end{aligned}$$

and

$$\zeta_d\big(\overline{x'_d},\ \iota_d(\overline{\mathbf{x}_{d-1}}, \overline{u})\big) = \langle \overline{x_d}, \overline{u}, \overline{\mathbf{x}_{d-1}}\rangle. \qquad \text{| Property (II)}$$

Therefore,

$$\zeta\big(\langle \overline{\mathbf{x}'_{d-1}}, \overline{x'_d}\rangle, \hat{\overline{u}}\big) = \langle \overline{\mathbf{x}_{d-1}}, \overline{x_d}, \overline{u}\rangle,$$

as required by Property (II). □

Definition 210. Dynamics D and D' are *equivalent* if (i) for every system S with dynamics D, there is a system S' with dynamics D' such that S and S' can implement the same functions, and (ii) for every system S' with dynamics D', there is a system S with dynamics D such that S and S' can implement the same functions.

Proposition 34. *Given η-finite dynamics D and D', if $\mathcal{C}_\eta(D) = \mathcal{C}_\eta(D')$, then D and D' are equivalent.*

Proof. Immediate by Theorem 31 of [37]. □

Lemma 17. *Let Φ be a class of functions satisfying the universal approximation property. Let us consider a cascade*

$$C = \big(\phi_{1,1} \circ D_{1,1} \parallel \cdots \parallel \phi_{1,m_1} \circ D_{1,m_1}\big) \ltimes \cdots \ltimes \big(\phi_{d,1} \circ D_{d,1} \parallel \cdots \parallel \phi_{d,m_d} \circ D_{d,m_d}\big)$$

where each $D_{i,j}$ is over $\mathbb{R}$, it is η-finite, and it has an input space U_i such that every η-component U^η of U_i satisfies $\bar{B}_{\Omega_{U_i}}(u, \epsilon) \subseteq U^\eta$ for some $u \in U^\eta$ and some $\epsilon > 0$. Then, there exist $\hat{\phi}_{i,j} \in \Phi$ such that the cascade

$$\hat{C} = \big(\hat{\phi}_{1,1} \circ D_{1,1} \parallel \cdots \parallel \hat{\phi}_{1,m_1} \circ D_{1,m_1}\big) \ltimes \cdots \ltimes \big(\hat{\phi}_{d,1} \circ D_{d,1} \parallel \cdots \parallel \hat{\phi}_{d,m_d} \circ D_{d,m_d}\big),$$

is equivalent to C.

Proof. Let $i \in [1..d]$ and let $j \in [1..m_i]$. Let U_i be the input space of $D_{i,j}$, and let $U^\eta_{i,1}, \ldots, U^\eta_{i,n_i}$ be the η-components of U_i. By assumption, for each $\ell \in [1..n_i]$, there are $\tilde{u}_{i,\ell} \in U^\eta_{i,\ell}$ and $\epsilon > 0$ such that $\bar{B}_{\Omega_{U_i}}(\tilde{u}_{i,\ell}, \epsilon) \subseteq U^\eta_{i,\ell}$. Hence, by Lemma 22 of [37], there is a continuous function $\phi'_{i,j}$ with image in $\{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,n_i}\}$ satisfying $\overline{\phi'_{i,j}} = \overline{\phi_{i,j}}$. We choose $\hat{\phi}_{i,j} \in \Phi$ that is an ϵ-approximation of $\phi'_{i,j}$. It follows that $\hat{\phi}_{i,j}(z) \in \bar{B}_{\Omega_{U_i}}(\phi'_{i,j}(z), \epsilon) \subseteq U^\ell_{i,j}$ for all $z \in U_i$, hence $\overline{\hat{\phi}_{i,j}} = \overline{\phi_{i,j}}$, and hence the η-canonical semiautomaton $\mathcal{C}_\eta(\hat{D}_{i,j})$ is the same as the η-canonical semiautomaton $\mathcal{C}_\eta(D_{i,j})$. Then, the same holds for the entire cascade; namely, the η-canonical semiautomaton $\mathcal{C}_\eta(\hat{C})$ coincides with the η-canonical semiautomaton $\mathcal{C}_\eta(C)$. Then the result follows immediately by Proposition 34. □

I.2 Realising semiautomata using single MinMax Recurrent Units

In this section we first provide an additional result and then provide full proofs of the results of Section 4.1. However, here we adopt a different execution of the proofs, where we first show the existence of MinMax Units that allow for choosing states and inputs in a flexible way, showing tha the input functions can be approximated, but replacing them with MLPs at a later step.

The next theorem is an additional result, showing that a single MinMax Recurrent Unit can realise any semiautomaton. The idea is to represent the transition matrix of the semiautomaton.

Lemma 18. *Let A be a semiautomaton having n states and m inputs, let $\epsilon \geq 0$, let $x_0, x_1 \in \mathbb{R}$ with $(x_1 - x_0) > 2\epsilon$, and let $U \subseteq \mathbb{R}$ be the union of m pair-wise disjoint closed intervals. There exist continuous functions $R : U \to \{x_0, x_1\}^{n \times n}$ and $s : U \to \{x_0, x_1\}^n$ such that semiautomaton A can be realised by the η-finite MinMax Recurrent Unit $D = \langle X^n, U, f \mid R, s\rangle$ where $X = [x_0 - \epsilon, x_0 + \epsilon] \cup [x_1 - \epsilon, x_1 + \epsilon]$; furthermore, for every ϵ-approximations $(\hat{R}, \hat{s})$ of (R, s), the MinMax Recurrent Unit $\hat{D} = \langle X^n, U, f \mid \hat{R}, \hat{s}\rangle$ is strongly equivalent to D.*

Proof. Let $A = \langle Q, \Sigma, \delta\rangle$ be a semiautomaton with states $Q = \{q_1, \ldots, q_n\}$ and input alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$.

Construction. We define the MinMax Recurrent Unit $D = \langle X^n, U, f \mid R, s\rangle$.

Let $U_1, \ldots, U_m \subseteq \mathbb{R}$ be the pair-wise disjount closed intervals such that $U = \bigcup_{i=1}^m U_i$.

We define the reset function $R : U \to \{x_0, x_1\}^{n \times n}$ and the set function $s : U \to \{x_0, x_1\}^n$. Let $\Delta := x_1 - x_0$. For each $i \in [1..m]$, let $T_i \in \{0, 1\}^{n \times n}$ be the matrix such that $(T_i)_{j,k} = 1$ iff $\delta(q_k, \sigma_i) = q_j$. For each $i \in [1..m]$, let $a_i : U \to \{0, \Delta\}$ be the function defined as $a_i(u) = \Delta$ iff $u \in U_i$. Note that the functions $a_1, \ldots, a_m$ are continuous by Lemma 2 since they are constant over η- components of U. Then,

$$R(u) := x_0 \cdot \mathbf{1}_{n,n} + \sum_{i=1}^{m} a_i(u) \cdot T_i,$$
$$s(u) := x_0 \cdot \mathbf{1}_n.$$

Note that function R is continuous since it is the composition of continuous functions, and function s is continuous since it is constant. This completes the construction of the MinMax Recurrent Unit. Next we prove the claims.

η-finiteness. The MinMax Recurrent Unit D is η-finite since U is a finite union of closed intervals, and X^n is the cross-product of a finite union of closed intervals.

Realisation. To show that D can realise A, it suffices to show that $\mathcal{M}(\mathcal{C}_\eta(D))$ is a realisation of $\mathcal{M}(A)$. Note that $\mathcal{M}(A) = \langle Q, \Sigma, \delta, \Gamma, \theta\rangle$ where $\Gamma = Q \times \Sigma$ and $\theta = \mathrm{id}$.

For each $i \in [1..m]$, let us choose an arbitrary $u_i \in \overline{U_i}$. The construction of D allows us to easily derive that its canonical semiautomaton has the form $\mathcal{C}_\eta(D) = \langle \overline{X}^n, \overline{U}, \overline{f} \rangle$ where

$$\overline{X}^n = \big\{ \langle \overline{b_1}, \ldots, \overline{b_n} \rangle \mid b_i \in \{x_0, x_1\} \big\},$$

$$\overline{U} = \{ \overline{u_1}, \ldots, \overline{u_m} \},$$

$$\overline{f}(\langle \overline{b_1}, \ldots, \overline{b_n} \rangle, \overline{u}) = \overline{f(\langle b_1, \ldots, b_n \rangle, u)}, \qquad \forall\, b_1, \ldots, b_n \in \{x_0, x_1\},\ \forall\, u \in U.$$

Then $\mathcal{M}(\mathcal{C}_\eta(D)) = \langle \overline{X}^n, \overline{U}, \overline{f}, \overline{X}^n \times \overline{U}, \mathrm{id} \rangle$.

To show that $\mathcal{M}(\mathcal{C}_\eta(D))$ is a realisation of $\mathcal{M}(A)$, it suffices to show an assignment (α, ι, ζ) of $\mathcal{M}(\mathcal{C}_\eta(D))$ into $\mathcal{M}(A)$, satisfying the two conditions (I) and (II) below for every $q \in Q$ and every $\sigma \in \Sigma$:

$$\text{(I)} \quad \overline{f}\big(\alpha(q), \iota(\sigma)\big) = \alpha\big(\delta(q, \sigma)\big),$$
$$\text{(II)} \quad \zeta\big(\alpha(q), \iota(\sigma)\big) = \langle q, \sigma \rangle.$$

Let us define the candidate assignment (α, ι, ζ) where

$$\alpha(q_i) := \langle \overline{b_1}, \ldots, \overline{b_n} \rangle \text{ with } \overline{b_j} = \overline{x_1} \text{ iff } j = i,$$

$$\iota(\sigma_i) := \overline{u_i},$$

$$\zeta\left(\langle \overline{b_1}, \ldots, \overline{b_n} \rangle, \overline{u_i}\right) := \begin{cases} \langle \alpha^{-1}(\overline{b_1}, \ldots, \overline{b_n}), \iota^{-1}(\overline{u_i}) \rangle & \text{if } \langle \overline{b_1}, \ldots, \overline{b_n} \rangle \in \mathrm{Im}(\alpha), \\ \langle q_1, \sigma_1 \rangle & \text{otherwise.} \end{cases}$$

Note that the inverse function α^{-1} is well-defined over the image $\mathrm{Im}(\alpha)$ since α is injective (intuitively, each tuple $\langle \overline{b_1}, \ldots, \overline{b_n} \rangle$ is the one-hot encoding of the index of a state q_i, with $\overline{x_1}$ standing for one and $\overline{x_0}$ standing for zero). Furthermore, the inverse function ι^{-1} is well-defined for all $\overline{u_i} \in \overline{U}$ since ι is a bijection.

Let $q_i \in Q$, let $\langle \overline{b_1}, \ldots, \overline{b_n} \rangle = \alpha(q_i)$, and let $\sigma_j \in \Sigma$.

We have that property (II) holds immediately, since by definition of ζ we have $\zeta\big(\alpha(q_i), \iota(\sigma_j)\big) = \zeta\big(\alpha^{-1}(\alpha(q_i)), \iota^{-1}(\iota(\sigma_j))\big) = \zeta\big(q_i, \sigma_j\big)$.

Next we show (I).

Note that $\overline{b_i} = \overline{x_1}$ and $\overline{b_\ell} = \overline{x_0}$ for every $\ell \neq i$. Let q_k be the next state of A, namely $q_k = \delta(q_i, \sigma_j)$. Thus $\alpha(\delta(q_i, \sigma)) = \langle \overline{b'_1}, \ldots, \overline{b'_n} \rangle$ with $\overline{b'_k} = \overline{x_1}$, and $\overline{b'_\ell} = \overline{x_0}$ for every $\ell \neq k$. To show (I) it suffices to show $\overline{f}(\langle \overline{b_1}, \ldots, \overline{b_n} \rangle, \overline{u_j}) = \langle \overline{b'_1}, \ldots, \overline{b'_n} \rangle$. In turn, letting $\ell \in [1..n]$, it suffices to show $(\overline{f}(\langle \overline{b_1}, \ldots, \overline{b_n} \rangle, \overline{u_j}))_\ell = \overline{x_1}$ if and only if $\ell = k$.

First, for every $p \in [1..n]$, we have

$$\big(R(u_j)\big)_{\ell,p} \;=\; x_0 + \Delta \cdot T_j \;=\; \begin{cases} x_0 + (x_1 - x_0) & \text{if } \delta(q_p, \sigma_j) = q_\ell, \\ x_0 & \text{otherwise,} \end{cases} \;=\; \begin{cases} x_1 & \text{if } \delta(q_p, \sigma_j) = q_\ell, \\ x_0 & \text{otherwise.} \end{cases}$$

Then,

$$\begin{aligned} \big(R(u_j) \otimes (b_1, \ldots, b_n)\big)_\ell &= \bigoplus_{p=1}^{n} \Big(\big(R(u_j)\big)_{\ell,p} \odot b_p \Big) \\ &= \Big(\big(R(u_j)\big)_{\ell,i} \odot x_1 \Big) \oplus \bigoplus_{p \neq i} \Big(\big(R(u_j)\big)_{\ell,p} \odot x_0 \Big) \\ &= \Big(\big(R(u_j)\big)_{\ell,i} \odot x_1 \Big) \oplus x_0 \\ &= \Big(\big(R(u_j)\big)_{\ell,i} \odot x_1 \Big) \\ &= \begin{cases} x_1 \odot x_1 & \text{if } \ell = k, \\ x_0 \odot x_1 & \text{otherwise,} \end{cases} \\ &= \begin{cases} x_1 & \text{if } \ell = k, \\ x_0 & \text{otherwise.} \end{cases} \end{aligned}$$

Then,

$$\begin{aligned}\big(f((b_1,\dots,b_n),u_j)\big)_\ell &= \big(f((b_1,\dots,b_n),u_j)\big)_\ell \\ &= \big((R(u_j)\otimes(b_1,\dots,b_n))\big)_\ell \oplus \big(s(u_j)\big)_\ell \\ &= \big((R(u_j)\otimes(b_1,\dots,b_n))\big)_\ell \oplus x_0 \\ &= \big(R(u_j)\otimes(b_1,\dots,b_n)\big)_\ell \\ &= \begin{cases} x_1 & \text{if } \ell = k, \\ x_0 & \text{otherwise.}\end{cases}\end{aligned}$$

Therefore,

$$\big(\overline{f}((\overline{b_1},\dots,\overline{b_n}),\overline{u_j})\big)_\ell = \big(\overline{f((b_1,\dots,b_n),u_j)}\big)_\ell = \begin{cases}\overline{x_1} & \text{if } \ell = k, \\ \overline{x_0} & \text{otherwise,}\end{cases}$$

as required.

Strong equivalence of approximations. For arbitrary $x \in X^n$ and $u \in U$, it suffices to show $\overline{f}(\overline{x},\overline{u}) = \overline{\hat{f}}(\overline{x},\overline{u})$. First we show $\overline{R}(u) = \overline{\hat{R}}(u)$ and $\overline{s}(u) = \overline{\hat{s}}(u)$. For every $\ell, p \in [1..n]$, we have $(\overline{R(u)})_{\ell,p} = (\overline{\hat{R}(u)})_{\ell,p}$ since $(R(u))_{\ell,p} \in \{x_0, x_1\}$, the intervals $[x_0-\epsilon, x_0+\epsilon]$ and $[x_1-\epsilon, x_1+\epsilon]$ are path-connected components, and $(R(u))_{\ell,p} = x_i$ implies $(\hat{R}(u))_{\ell,p} \in [x_i-\epsilon, x_i+\epsilon]$. Similarly, for every $\ell \in [1..n]$, we have $(\overline{s(u)})_\ell = (\overline{\hat{s}(u)})_\ell$ since $(s(u))_\ell = x_0$, the interval $[x_0-\epsilon, x_0+\epsilon]$ is a η-component, and $(s(u))_\ell = x_0$ implies $(\hat{s}(u))_\ell \in [x_0-\epsilon, x_0+\epsilon]$. Then, recalling the notation for MinMax recurrence (Definition 200), $\overline{f}(\overline{x},\overline{u}) = \overline{f(x,u)} = \overline{\circledast(x, R(u), s(u))}$ and $\overline{\hat{f}}(\overline{x},\overline{u}) = \overline{\hat{f}(x,u)} = \overline{\circledast(x,\hat{R}(u),\hat{s}(u))}$. Having $\overline{R}(u) = \overline{\hat{R}}(u)$ and $\overline{s}(u) = \overline{\hat{s}}(u)$, we obtain $\overline{\circledast(x,\hat{R}(u),\hat{s}(u))} = \overline{\circledast(x,R(u),s(u))}$ by continuity of the $\circledast$ function, since the continuous image of path-connected components is path-connected. Therefore, $\overline{f}(\overline{x},\overline{u}) = \overline{\hat{f}}(\overline{x},\overline{u})$ as required. □

The next lemma is the full version of Theorem 3 from Section 4.1, except that we postpone replacing the input functions with MLPs.

Lemma 19. *Let A be an identity-reset semiautomaton having n states and m inputs, let $\epsilon \geq 0$, let $x_1,\dots,x_n \in \mathbb{R}$ with $|x_i - x_j| > 2\epsilon$ for all $i \neq j$, and let $U \subseteq \mathbb{R}$ be the union of m pair-wise disjoint closed intervals. There exist continuous functions $R, s : U \to \{x_1,\dots,x_n\}$ such that semiautomaton A can be realised by the η-finite MinMax Recurrent Unit $D = \langle X, U, f \mid R, s\rangle$ where $X = \bigcup_{i=1}^n [x_i-\epsilon, x_i+\epsilon]$; furthermore, for every ϵ-approximations $(\hat{R},\hat{s})$ of (R,s), the MinMax Recurrent Unit $\hat{D} = \langle X, U, f \mid \hat{R}, \hat{s}\rangle$ is strongly equivalent to D.*

Proof. Let $A = \langle Q, \Sigma, \delta\rangle$ be an identity-reset semiautomaton with states $Q = \{q_1,\dots,q_n\}$ and inputs $\Sigma = \{\sigma_1,\dots,\sigma_m\}$.

Construction. We define the MinMax Recurrent Unit $D = \langle X, U, f \mid R, s\rangle$. Let $U_1,\dots,U_m$ be the pair-wise disjoint closed intervals such that $U = \bigcup_{i=1}^m U_i$. Let $x_{\max} := \max\{x_1,\dots,x_n\}$, and let $x_{\min} := \min\{x_1,\dots,x_n\}$.

Next we define the input functions $R : U \to \{x_1,\dots,x_n\}$ and $s : U \to \{x_1,\dots,x_n\}$. For $u \in U_i$, letting $\tau_i(q) := \delta(q,\sigma_i)$ be the state transformation induced by σ_i, we define

$$R(u) := \begin{cases} x_{\min} & \text{if } \tau_i \text{ is identity,} \\ x_{\max} & \text{if } \tau_i \text{ is constant,}\end{cases}$$

$$s(u) := \begin{cases} x_{\min} & \text{if } \tau_i \text{ is identity,} \\ x_j & \text{if } \tau_i \text{ is constant with value } q_j.\end{cases}$$

Note that the functions R and s are continuous by Lemma 2 since they are constant on each η-component U_i.

This completes the construction of the MinMax Recurrent Unit. Next we prove the claims.

η-finiteness. The MinMax Recurrent Unit D is η-finite since U and X are finite unions of closed intervals.

Realisation. To show that D can realise A, it suffices to show that $\mathcal{M}(\mathcal{C}_\eta(D))$ is a realisation of $\mathcal{M}(A)$. Note that $\mathcal{M}(A) = \langle Q, \Sigma, \delta, \Gamma, \theta \rangle$ where $\Gamma = Q \times \Sigma$ and $\theta = \mathrm{id}$.

For each $i \in [1..m]$, let us choose an arbitrary $u_i \in \overline{U_i}$. The construction of D allows us to easily derive that its canonical semiautomaton has the form $\mathcal{C}_\eta(D) = \langle \overline{X}, \overline{U}, \overline{f} \rangle$ where

$$\begin{aligned} \overline{X} &= \{\overline{x}_1, \ldots, \overline{x}_n\}, \\ \overline{U} &= \{\overline{u}_1, \ldots, \overline{u}_m\}, \\ \overline{f}(\overline{x}, \overline{u}) &= \overline{f(x,u)}, \qquad\qquad \forall\, x \in \{x_1, \ldots, x_n\},\ \forall\, u \in U. \end{aligned}$$

Then $\mathcal{M}(\mathcal{C}_\eta(D)) = \langle \overline{X}, \overline{U}, \overline{f}, \overline{X} \times \overline{U}, \mathrm{id} \rangle$.

To show that $\mathcal{M}(\mathcal{C}_\eta(D))$ is a realisation of $\mathcal{M}(A)$, it suffices to show an assignment (α, ι, ζ) of $\mathcal{M}(\mathcal{C}_\eta(A))$ into $\mathcal{M}(A)$, satisfying the two conditions (I) and (II) below for every $q_i \in Q$ and every $\sigma_j \in \Sigma$.

$$\begin{aligned} &\text{(I)} \quad \overline{f}\big(\alpha(q_i), \iota(\sigma_j)\big) = \alpha\big(\delta(q_i, \sigma_j)\big) \\ &\text{(II)} \quad \zeta\big(\alpha(q_i), \iota(\sigma_j)\big) = \theta\big(q_i, \sigma_j\big) \end{aligned}$$

Let us define the candidate assignment (α, ι, ζ) where

$$\begin{aligned} \alpha(q_i) &:= \overline{x_i}, \\ \iota(\sigma_j) &:= \overline{u_j}, \\ \zeta(\overline{x_i}, \overline{u_j}) &:= \langle q_i, \sigma_j \rangle. \end{aligned}$$

Let $q_i \in Q$, let $\sigma_j \in \Sigma$, and let us note that $\alpha(q_i) = \overline{x_i}$ and $\iota(\sigma_j) = \overline{u_j}$. We have that condition (II) holds by definition of ζ.

Next we show (I). We have

$$\overline{f}\big(\alpha(q_i), \iota(\sigma_j)\big) = \overline{f}\big(\overline{x_i}, \overline{u_j}\big) = \overline{f\big(x_i, u_j\big)}$$

Let $\tau(q) := \delta(q, \sigma_j)$. Let q_k be the next state of A, namely $q_k = \tau(q_i)$. As $\alpha(\delta(q_i, \sigma_j)) = x_k$, to show (I), it suffices to show $f(x_i, u_j) = x_k$.

We have that τ is a reset-identity transformation, and hence it is either identity or constant.

Say that τ is identity, hence $q_k = q_i$, and hence we need to show $f(x_i, u_j) = x_i$. We have

$$\begin{aligned} f(x_i, u_j) &= (x_i \odot x_{\max}) \oplus x_{\min} && \text{by def. of } R(u_j), \\ &= x_i \oplus x_{\min} && \text{since } x_i \leq x_{\max} \text{ by construction of } x_{\max}, \\ &= x_i && \text{since } x_i \geq x_{\min} \text{ by construction of } x_{\min}, \end{aligned}$$

which proves the case when τ is identity.

Now say that τ is constant with value q_k, and hence it suffices to show $f(x_i, u_j) = x_k$. We have

$$\begin{aligned} f(x_i, u_j) &= (x_i \odot x_{\min}) \oplus x_k && \text{by def. of } R(u_j), \\ &= x_{\min} \oplus x_k && \text{since } x_i \geq x_{\min} \text{ by construction of } x_{\min}, \\ &= x_k && \text{since } x_k \leq x_{\max} \text{ by construction of } x_{\max}, \end{aligned}$$

which proves the case when τ is constant.

This concludes the proof of (I), and hence the proof that the MinMax Recurrent Unit can realise A.

Strong equivalence of approximations. Omitted since it is analogous to the one of Lemma 18. □

The next lemma is the full version of Theorem 4 from Section 4.1, except that we postpone replacing the input functions with MLPs.

Lemma 20. *Let A be a permutation semiautomaton having n inputs and permutation degree d. Let $\epsilon \geq 0$, let $x_1, \ldots, x_d \in \mathbb{R}$ with $|x_i - x_j| > 2\epsilon$ for all $i \neq j$, and let $U \subseteq \mathbb{R}$ be the union of n pair-wise disjoint closed intervals. There exist continuous functions $R : U \to \{x_1, \ldots, x_d\}^{d \times d}$ and $s : U \to \{x_1, \ldots, x_d\}^d$ such that semiautomaton A can be realised by the η-finite MinMax Recurrent Unit $D = \langle X^d, U, f \mid R, s\rangle$ where $X = \bigcup_{i=1}^{d}[x_i - \epsilon, x_i + \epsilon]$; furthermore, for every ϵ-approximations $(\hat{R}, \hat{s})$ of (R, s), the MinMax Recurrent Unit $\hat{D} = \langle X^d, U, f \mid \hat{R}, \hat{s}\rangle$ is strongly equivalent to D.*

Proof. Since A has permutation degree d, by Theorem 22, there exists $\varphi : \Sigma \to \mathrm{Sym}([1..d])$ such that the semiautomaton $A' = \langle [1..d]^d, \Sigma, \delta'\rangle$ with $\delta'(\langle q_1, \ldots, q_d\rangle, \sigma) = \langle q_{\pi(1)}, \ldots, q_{\pi(n)}\rangle$ for $\pi : [1..d] \to [1..d]$ the permutation given by $\pi = \varphi(\sigma)$. Let $U_1, \ldots, U_n$ be the pair-wise disjoint closed intervals such that $U = \bigcup_{i=1}^{n} U_i$. Let us enumerate the elements of Σ as $\sigma_1, \ldots, \sigma_n$. Let π_i be the permutation given by $\pi_i = \varphi(\sigma_i)$.

Construction. We define the MinMax Recurrent Unit $D = \langle X, U, f \mid R, s\rangle$. Let $x_{\max} = \max\{x_1, \ldots, x_d\}$, let $x_{\min} = \min\{x_1, \ldots, x_d\}$, and let $x_\Delta = x_{\max} - x_{\min}$.

For each $i \in [1..n]$, let $P_i \in \{0, 1\}^{n \times n}$ be the (permutation) matrix where $(P_i)_{j,k} = 1$ iff $\pi_i(k) = j$, and let $a_i : U \to \{0, x_\Delta\}$ be the function defined as $a_i(u) = x_\Delta$ iff $u \in U_i$. Then, we define the functions $R : U \to \{x_1, \ldots, x_d\}^{d \times d}$ and $s : U \to \{x_1, \ldots, x_d\}^d$ as

$$R(u) := x_{\min} \cdot \mathbf{1}_{d,d} + \sum_{i=1}^{n} a_i(u) \cdot P_i \qquad s(u) := x_{\min} \cdot \mathbf{1}_d.$$

Note that each function a_i is continuous since it is constant over η-components of U, and hence R is continuous since it is a finite composition of continuous functions. Also s is continuous since it is constant. This completes the construction of the MinMax Recurrent Unit. Next we prove the claims.

η-finiteness. The MinMax Recurrent Unit D is η-finite since U is a finite union of closed intervals, and X is the cross-product of a finite union of closed intervals.

Realisation. To show that D can realise A, it suffices to show that D can realise A' since A' realises A. In turn, it suffices to show that $\mathcal{M}(\mathcal{C}_\eta(D))$ is a realisation of $\mathcal{M}(A')$. Note that $\mathcal{M}(A') = \langle [1..d]^d, \Sigma, \delta', [1..d]^d \times \Sigma, \mathrm{id}\rangle$.

The construction of D allows us to easily derive that its canonical semiautomaton has the form $\mathcal{C}_\eta(D) = \langle \overline{X}^d, \overline{U}, \overline{f}\rangle$ where

$$\overline{X}^d = \{\langle \overline{x_{i_1}}, \ldots, \overline{x_{i_d}}\rangle \mid i_1, \ldots, i_d \in [1..d]\},$$
$$\overline{U} = \{\overline{u_i} \mid i \in [1..n]\},$$
$$\overline{f}(\langle \overline{x_{i_1}}, \ldots, \overline{x_{i_d}}\rangle, \overline{u_i}) = \overline{f(\langle x_{i_1}, \ldots, x_{i_d}\rangle, u_i)}$$

Thus $\mathcal{M}(\mathcal{C}_\eta(D)) = \langle \overline{X}^d, \overline{U}, \overline{f}, \overline{X}^d \times \overline{U}, \mathrm{id}\rangle$.

To show that $\mathcal{M}(\mathcal{C}_\eta(D))$ is a realisation of $\mathcal{M}(A')$, it suffices to show an assignment (α, ι, ζ) of $\mathcal{M}(\mathcal{C}_\eta(D))$ into $\mathcal{M}(A')$, satisfying the two conditions (I) and (II) below for every $\langle q_1, \ldots, q_d\rangle \in [1..d]^d$ and every $\sigma \in \Sigma$, letting $\langle \overline{x_{q_1}}, \ldots, \overline{x_{q_d}}\rangle = \alpha(q_1, \ldots, q_d)$,

$$\text{(I)} \quad \overline{f}\big(\langle \overline{x_{q_1}}, \ldots, \overline{x_{q_d}}\rangle, \iota(\sigma)\big) = \alpha\big(\delta'(\langle q_1, \ldots, q_d\rangle, \sigma)\big)$$
$$\text{(II)} \quad \zeta\big(\langle \overline{x_{q_1}}, \ldots, \overline{x_{q_d}}\rangle, \iota(\sigma)\big) = \langle \langle q_1, \ldots, q_d\rangle, \sigma\rangle$$

Let us define the candidate assignment (α, ι, ζ) where

$$\alpha\left(\langle q_1, \ldots, q_d\rangle\right) := \langle \overline{x_{q_1}}, \ldots, \overline{x_{q_1}}\rangle,$$
$$\iota(\sigma_i) := \overline{u_i},$$
$$\zeta\left(\langle \overline{x_{q_1}}, \ldots, \overline{x_{q_d}}\rangle, \overline{u_i}\right) := \langle \langle q_1, \ldots, q_d\rangle, \sigma_i\rangle.$$

Let $\langle q_1, \dots, q_d \rangle \in [1..d]^d$ and let $\sigma_i \in \Sigma$. Note $\alpha(q_1, \dots, q_d) = \langle \overline{x_{q_1}}, \dots, \overline{x_{q_d}} \rangle$ and $\iota(\sigma_i) = \overline{u_i}$. Condition (II) holds immediately by construction. Next we show condition (I).

We have

$$\delta'(\langle q_1, \dots, q_d \rangle, \sigma_i) = \langle q_{\pi_i(1)}, \dots, q_{\pi_i(d)} \rangle,$$

and hence letting $j \in [1..d]$ and $j' = \pi_i(j)$, by the definition of α it suffices to show

$$\Big(\overline{f}\big(\langle \overline{x_{q_1}}, \dots, \overline{x_{q_d}} \rangle, \overline{u_i}\big)\Big)_j = \overline{x_{q_{j'}}}.$$

First, we have

$$R(u_i) = x_{\min} \cdot \mathbf{1}_{n,n} + x_\Delta \cdot P_i,$$
$$s(u_i) = x_{\min} \cdot \mathbf{1}_n,$$

and hence, for every $\ell \in [1..d]$, we have

$$\big(R(u_i)\big)_{j,\ell} = x_{\min} + x_\Delta \cdot \big(P_i\big)_{j,\ell} = \begin{cases} x_{\max} & \text{if } \pi_i(j) = \ell, \\ x_{\min} & \text{otherwise,} \end{cases} = \begin{cases} x_{\max} & \text{if } \ell = j', \\ x_{\min} & \text{otherwise,} \end{cases}$$
$$\big(s(u_i)\big)_j = x_{\min}.$$

Thus,

$$\begin{aligned}
\Big(f\big(\langle x_{q_1}, \dots, x_{q_d} \rangle, u_i\big)\Big)_j &= \left(\bigoplus_{\ell=1}^{d} \big(R(u_i)\big)_{j,\ell} \odot x_{q_\ell}\right) \oplus \big(s(u_i)\big)_j \\
&= (x_{\max} \odot x_{q_{j'}}) \oplus \left(\bigoplus_{\ell \neq j'} x_{\min} \odot x_{q_\ell}\right) \oplus \big(s(u_i)\big)_j \\
&= x_{q_{j'}} \oplus \left(\bigoplus_{\ell \neq j'} x_{\min}\right) \oplus \big(s(u_i)\big)_j \\
&= x_{q_{j'}} \oplus x_{\min} \oplus \big(s(u_i)\big)_j \\
&= x_{q_{j'}} \oplus x_{\min} \oplus x_{\min} \\
&= x_{q_{j'}},
\end{aligned}$$

and hence

$$\Big(\overline{f}\big(\langle \overline{x_{q_1}}, \dots, \overline{x_{q_d}} \rangle, \overline{u_i}\big)\Big)_j = \Big(\overline{f\big(\langle \overline{x_{q_1}}, \dots, \overline{x_{q_d}} \rangle, \overline{u_i}\big)}\Big)_j = \overline{x_{q_{j'}}},$$

as required. This proves that (α, ι, ζ) is an assignment of $\mathcal{M}(A')$ into $\mathcal{M}(\mathcal{C}_\eta(D))$, and hence overall we have proved that D can realise A, by the arguments put forward in the course of the proof.

Strong equivalence of approximations. Omitted since it is analogous to the one of Lemma 18. □

I.3 Realising Cascades

This section develops part of the proof of Theorem 1. As discussed in the proof sketch of the theorem, the expressivity results are obtained via realising cascades of semiautomata. In this section we show that the MinMax Units obtained in the previous section can be composed into cascade to realise cascades of semiautomata.

Theorem 23. *Let us consider a cascade*

$$C = \big(A_{1,1} \parallel \cdots \parallel A_{1,m_1}\big) \ltimes \cdots \ltimes \big(A_{d,1} \parallel \cdots \parallel A_{d,m_d}\big),$$

where each $A_{i,j}$ is a semiautomaton. We have that C can be realised by the following η-finite MinMax Recurrent Cascade

$$C' = \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big),$$

where the input functions of each unit $D_{i,j}$ are MLPs, and the state dimension of each unit $D_{i,j}$ is (i) one if $A_{i,j}$ is identity-reset, (ii) the degree of $A_{i,j}$ if $A_{i,j}$ is a permutation semiautomaton, (iii) the number of states of $A_{i,j}$ otherwise.

Proof. We have that each $A_{i,j}$ can be realised by an η-finite MinMax Recurrent Unit $D'_{i,j}$ with MLPs as input functions and state dimension as required. In fact, (i) when $A_{i,j}$ is identity-reset, we can apply Theorem 19; (ii) when $A_{i,j}$ is permutation, we can apply Theorem 20; (iii) we can always apply Theorem 18.

Then, by Lemma 16, we have that C can be realised by a cascade

$$C' = \big(\phi_{1,1} \circ D'_{1,1} \parallel \cdots \parallel \phi_{1,m_1} \circ D'_{1,m_1}\big) \ltimes \cdots \ltimes \big(\phi_{d,1} \circ D'_{d,1} \parallel \cdots \parallel \phi_{d,m_d} \circ D'_{d,m_d}\big),$$

where each $\phi_{i,j}$ is a continuous function, and each $\phi_{1,j}$ is over a compact subset $\hat{U}$ of $\mathbb{R}$; and we note that each $\phi_{i,j}$ with $i > 1$ is also over a compact real-valued domain, as its domain is given by the cross-product of U and η-finite state spaces, which are compact by definition. Then, by Proposition 33 and by the Universal Approximation Theorem for MLPs, we have that each $\phi_{i,j} \circ D'_{i,j}$ is strongly equivalent to $\psi_{i,j} \circ D'_{i,j}$ for some MLP $\psi_{i,j}$. Thus, the following cascade

$$\hat{C}' = \big(\psi_{1,1} \circ D'_{1,1} \parallel \cdots \parallel \psi_{1,m_1} \circ D'_{1,m_1}\big) \ltimes \cdots \ltimes \big(\psi_{d,1} \circ D'_{d,1} \parallel \cdots \parallel \psi_{d,m_d} \circ D'_{d,m_d}\big)$$

is strongly equivalent to C'; namely, $\mathcal{C}_\eta(C') = \mathcal{C}_\eta(\hat{C}')$. Hence, $\hat{C}'$ can realise C, since C' can realise C. Then, we note that each $\hat{D}'_{i,j} := \psi_{i,j} \circ D'_{i,j}$ is a MinMax Recurrent Unit, since $D'_{i,j}$ is so, and composing it with $\psi_{i,j}$ amounts to composing the input functions of $D'_{i,j}$ with $\psi_{i,j}$. Therefore, $\hat{C}'$ is a MinMax RNC and the given cascade C can be realised by $\hat{C}'$. □

The following theorem regards floating-point MinMax RNCs, that are defined in Section H.3.2.

Theorem 24. *Let us consider a cascade*

$$C = \big(A_{1,1} \parallel \cdots \parallel A_{1,m_1}\big) \ltimes \cdots \ltimes \big(A_{d,1} \parallel \cdots \parallel A_{d,m_d}\big),$$

where each $A_{i,j}$ is a semiautomaton. We have that C can be realised by a floating-point MinMax Recurrent Cascade over some $\mathbb{F}$, having the form

$$C' = \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big),$$

where the input functions of each unit $D_{i,j}$ are $\mathbb{F}$-MLPs, and the state dimension of each unit $D_{i,j}$ is (i) one if $A_{i,j}$ is identity-reset, (ii) the degree of $A_{i,j}$ if $A_{i,j}$ is a permutation semiautomaton, (iii) the number of states of $A_{i,j}$ otherwise.

Proof. For each $A_{i,j}$, let us define $m_{i,j}$ as follows:

- if $A_{i,j}$ is a permutation semiautomaton, then $m_{i,j}$ is the maximum between the degree and the number of inputs of $A_{i,j}$;
- otherwise, $m_{i,j}$ is the maximum between the number of states and the number of inputs of $A_{i,j}$.

Then, let m be the largest $m_{i,j}$, and let $\mathbb{F}$ be a floating-point domain containing at least m elements.

We have that each $A_{i,j}$ can be realised by a MinMax Recurrent Unit $D'_{i,j}$ over $\mathbb{F}$ with $\mathbb{F}$-MLPs as input functions and state dimension as required. In fact, (i) when $A_{i,j}$ is identity-reset, we apply Theorem 19; (ii) when $A_{i,j}$ is permutation, we apply Theorem 20; (iii) otherwise, we apply Theorem 18.

Then, by Lemma 16, we have that C can be realised by a cascade

$$C' = \big(\phi_{1,1} \circ D'_{1,1} \parallel \cdots \parallel \phi_{1,m_1} \circ D'_{1,m_1}\big) \ltimes \cdots \ltimes \big(\phi_{d,1} \circ D'_{d,1} \parallel \cdots \parallel \phi_{d,m_d} \circ D'_{d,m_d}\big),$$

where each $\phi_{1,j}$ has a domain $\hat{U} \subseteq \mathbb{F}^k$ for some k, and hence each $\phi_{i,j}$ is necessarily a function with $\mathbb{F}$-valued domain and codomain, since it maps from the cross product of $\hat{U}$ with state spaces of the Units $D'_{i,j}$ to the state space of one of the Units, which are all $\mathbb{F}$-valued. Thus, each $D_{i,j} := \phi_{i,j} \circ D'_{i,j}$ is a MinMax Recurrent Unit with input functions $\phi_{i,j} \circ R_{i,j}$ and each $\phi_{i,j} \circ s_{i,j}$ over $\mathbb{F}$, which can

be expressed as $\mathbb{F}$-MLPs by any of Corollary 3.8 or Corollary 3.11 of [60]. Therefore, C' can be equivalently rewritten as

$$C' = \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big),$$

which is as required. □

Theorem 25. *Let us consider a cascade*

$$C = \big(A_{1,1} \parallel \cdots \parallel A_{1,m_1}\big) \ltimes \cdots \ltimes \big(A_{d,1} \parallel \cdots \parallel A_{d,m_d}\big),$$

where each $A_{i,j}$ is a semiautomaton. We have that C can be realised by an η-finite MinMax Recurrent Cascade, and also by a floating-point MinMax Recurrent Cascade over some $\mathbb{F}$, both of the form

$$C' = \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big),$$

where the input functions of each unit $D_{i,j}$ are MLPs and $\mathbb{F}$-MLPs, respectively, and the state dimension of each unit $D_{i,j}$ is (i) one if $A_{i,j}$ is identity-reset, (ii) the degree of $A_{i,j}$ if $A_{i,j}$ is a permutation semiautomaton, (iii) the number of states of $A_{i,j}$ otherwise.

Proof. It holds immediately by Theorem 23 and Theorem 24, which focus on the real-valued and float-valued case, respectively. □

I.4 Formal Expressivity (Proof of Theorem 1)

In this section we conclude the proof of Theorem 1 putting together the results obtained in the previous sections. We prove it by parts, restating the theorem at the end and referring to all the proven sub-theorems.

The next theorem corresponds to the first claim in Theorem 1.

Theorem 26. *Every regular function F (regular language L) having degree d can be implemented (recognised) by an η-finite MinMax RNC and by a floating-point MinMax RNC, both having state degree d.*

Proof. We first note that the case of regular languages is subsumed by the case of regular functions. In fact, for every language L over an alphabet Σ, we have that a system can recognise L if it can implement its characteristic function $\mathbb{I}_L : \Sigma^* \to \{0, 1\}$, by definition, and such function is a regular function. Thus, we proceed with the proof for functions.

Let $F : \Sigma^* \to \Gamma$ be a regular function having degree n. let A_F be the canonical automaton of F and let D_F be the semiautomaton of A_F. By Theorem 12, semiautomaton D_F can be realised by a cascade

$$C = A_1 \ltimes \cdots \ltimes A_n$$

where each A_i is either a two-state identity-reset semiautomaton or a G-semiautomaton for G a simple group that is a factor of a subgroup of the transition monoid of D_F. We note that, since F has degree d by assumption, the permutation degree d_i of A_i is at most d.

The rest of the proof is split into two parts: in the first part we show the existence of an η-finite MinMax RNC witnessing the claim, and in the second part we show the existence of a floating-point MinMax RNC witnessing the claim—the arguments in the two parts are similar, but we keep the two parts separate for the sake of clarity.

Part I. By Theorem 25, we have that cascade C can be realised by a cascade

$$C' = D_1 \ltimes \cdots \times D_n$$

where each D_i is an η-finite MinMax Recurrent Unit with MLPs as input functions. Letting $C' = \langle X, U, f\rangle$, by Theorem 15, we have that C' can be extended into a system $S = \langle X, U, f, x_0, X \times U, \mathrm{id}\rangle$ that can implement F. In particular, system S has an identity output function, which can be seen as a trivial MLP. Therefore, S is an η-finite MinMax RNC that can implement F, as required.

Part II. By Theorem 24, we have that cascade C can be realised by a cascade

$$C' = D_1 \ltimes \cdots \times D_n$$

where each D_i is a floating-point MinMax Recurrent Unit over some floating-point domain $\mathbb{F}$ with $\mathbb{F}$-MLPs as input functions, both having state dimension equal to one if A_i is identity-reset and equal to the degree d_i of A_i otherwise; we note that $d_i \leq d$. Letting $C' = \langle X, U, f \rangle$, by Theorem 15, we have that C' can be extended into a system $S = \langle X, U, f, x_0, X \times U, \mathrm{id} \rangle$ that can implement F. In particular, system S has an identity output function, which can be seen as a trivial $\mathbb{F}$-MLP. Therefore, S is a floating-point MinMax RNC that can implement F. □

Next we show the following intermediate result in order to prove the second claim of Theorem 1.

Lemma 21. *Let Σ and Γ be alphabets, and let $S = \langle X, U, f, x^0, Y, g \rangle$ be a MinMax Recurrent Cascade. For every function $F : \Sigma^+ \to \Gamma$ that is implemented by S with encoder $\mathrm{enc} : \Sigma \to U$ and decoder $\mathrm{dec} : Y \to \Gamma$, there exist finite subsets $\tilde{X} \subseteq X$, $\tilde{U} \subseteq U$, $\tilde{Y} \subseteq Y$ such that the MinMax Recurrent Cascade $\tilde{S} = \langle \tilde{X}, \tilde{U}, f, x^0, \tilde{Y}, g \rangle$ implements F with the same encoder enc and decoder dec.*

Proof. Let us consider a finite input alphabet Σ and a finite output alphabet Γ. Let $S = \langle X, U, f, x^0, Y, g \rangle$ be a MinMax RNC, and let $F : \Sigma^+ \to \Gamma$ be a function that is implemented by S with encoder $\mathrm{enc} : \Sigma \to U$ and decoder $\mathrm{dec} : Y \to \Gamma$.

Since Σ is finite, the image of enc is a finite set $\tilde{U} := \{u_1, \ldots, u_m\} \subseteq U$.

Let the dynamics of S be of the form

$$C = \big(D_{1,1} \parallel \cdots \parallel D_{1,m_1}\big) \ltimes \cdots \ltimes \big(D_{d,1} \parallel \cdots \parallel D_{d,m_d}\big),$$

with $D_{i,j} = \langle X_{i,j}, U_i, f_{i,j} \mid R_{i,j}, s_{i,j} \rangle$. Letting $X_i := X_{i,1} \times \cdots \times X_{i,m_i}$, we have that $U_i = U \times X_1 \times \cdots \times X_{i-1}$, and the state space of C is $X = X_1 \times \cdots \times X_d$.

We show there exists a finite $\tilde{X}$ with $\{x^0\} \subseteq \tilde{X} \subseteq X$ such that, for every input string $\sigma_1 \cdots \sigma_\ell \in \Sigma^+$, we have that $C(x^0, \mathrm{enc}(\sigma_1 \cdots \sigma_\ell)) \in \tilde{X}$; it will follow that the states of C can be restricted to $\tilde{X}$, and that Y can be restricted to $\tilde{Y} := \{g(x, u) \in Y \mid x \in \tilde{X}, u \in \tilde{U}\}$.

For every $i \in [1..d]$ and every $j \in [1..m_i]$, let $n_{i,j}$ be the state dimension of $D_{i,j}$, and let us define

$$\tilde{X}^0_{i,j} := \Big\{ \big(x^0_{i,j}\big)_p \;\Big|\; p \in [1..n_{i,j}] \Big\}, \tag{38}$$

$$\tilde{X}_{i,j} := \tilde{X}^0_{i,j} \cup \Big\{ \big(R_{i,j}(u)\big)_{p,q}, s_{i,j}(u)\big)_p \;\Big|\; u \in \big(\tilde{U} \times \tilde{\mathbf{X}}_{i-1}\big),\, p, q \in [1..n_{i,j}] \Big\}, \tag{39}$$

$$\tilde{X}_i := \big(\tilde{X}_{i,1}\big)^{n_{i,1}} \times \cdots \big(\tilde{X}_{i,m_i}\big)^{n_{i,m_i}}, \tag{40}$$

$$\tilde{\mathbf{X}}_i := \tilde{X}_1 \times \cdots \tilde{X}_i, \tag{41}$$

$$\tilde{X} := \tilde{\mathbf{X}}_d. \tag{42}$$

For an arbitrary input string $\sigma_1, \ldots, \sigma_\ell \in \Sigma^*$, its encoding is in $\tilde{U}^*$.

We proceed by induction on d, showing that $\tilde{\mathbf{X}}_d$ includes the state of C upon reading any sequence in $\tilde{U}^*$.

In the base case $d = 1$. The input elements to the units $D_{1,j}$ are from $\tilde{U}$, and the the image of the dynamics function of $D_{1,j}$ consist of $n_{1,j}$-vectors with components that are necessarily (i) a component $\big(x^0_{1,j}\big)_p$ of its initial state $x^0_{1,j}$, (ii) an entry $\big(R_{1,j}(u)\big)_{p,q}$ of the matrix $R_{1,j}(u)$ returned by its reset function $R_{1,j}$ on an input $u \in \tilde{U}$, (iii) a component $\big(s_{1,j}(u)\big)_p$ of the vector $s_{1,j}(u)$ returned by its set function $s_{1,j}$ on an input $u \in \tilde{U}$. The three cases are included in the definition of $\tilde{X}_{i,j}$ (Eq. (39)), and hence the set of possible states of $D_{1,j}$ is contained in $\big(\tilde{X}_{1,j}\big)^{n_{1,j}}$. Then the set of possible states of the cascade (a single layer in this case) is contained in the cross-product $\tilde{X}_1$ (Eq. (40)), and we have $\tilde{\mathbf{X}}_1 = \tilde{X}_1$ (Eq. (41)).

In the inductive case $d > 1$, and we assume by induction that $\tilde{\mathbf{X}}_{d-1}$ includes the state of the first $d-1$ layers of C upon reading any sequence in $\tilde{U}^*$. Hence, the input elements to each unit $D_{d,j}$ are contained in the set $\tilde{U} \times \tilde{\mathbf{X}}_{d-1}$. The image of the dynamics function of $D_{d,j}$ consist of $n_{d,j}$-vectors with components that are necessarily (i) a component $\left(x^0_{d,j}\right)_p$ of its initial state $x^0_{d,j}$, (ii) an entry $\left(R_{d,j}(u)\right)_{p,q}$ of the matrix $R_{d,j}(u)$ returned by its reset function $R_{d,j}$ on an input $u \in \tilde{U} \times \tilde{\mathbf{X}}_{d-1}$, (iii) a component $\left(s_{d,j}(u)\right)_p$ of the vector $s_{d,j}(u)$ returned by its set function $s_{d,j}$ on an input $u \in \tilde{U} \times \tilde{\mathbf{X}}_{d-1}$. The three cases are included in the definition of $\tilde{X}_{d,j}$ (Eq. (39)), and hence the set of possible states of $D_{d,j}$ is contained in $\left(\tilde{X}_{d,j}\right)^{n_{d,j}}$. Then the set of possible states of layer d of the cascade is contained in the cross-product $\tilde{X}_d$ (Eq. (40)). By the inductive hypothesis, we obtain that the possible sates of C is contained in the cross-product $\tilde{\mathbf{X}}_d = \tilde{X}_1 \times \cdots \times \tilde{X}_d$ (Eq. (41)).

Then, by definition of $\tilde{X} = \tilde{\mathbf{X}}_d$ (Eq. (42)), we obtain that $\tilde{X}$ contains the possible sates of C. □

The next theorem corresponds to the second claim of Theorem 1.

Theorem 27. *Every function (language) that is not regular cannot be implemented (recognised) by a MinMax Recurrent Cascade.*

Proof. Let F be a function that can be implemented by a MinMax Recurrent Cascade S. By Lemma 21, F can also be implemented by a MinMax Recurrent Cascade S' that has a finite input and state space, and hence trivially η-finite. It follows that F is regular by Theorem 13. The proof applies to languages as well, since recognising a language means implementing its characteristic function. □

We proceed towards proving the third claim of Theorem 1. We employ the notions of η-convergence and aperiodicity from MAT, introduced in Section A.19.1.

Lemma 22. *Let $D = D_1 \parallel \cdots \parallel D_n$ be a parallel composition of η-finite aperiodic dynamics. Then D is aperiodic.*

Proof. We have that each D_i is of the form $D_i = \langle X_i, U, f_i \rangle$. Then, we have that D is of the form $D = \langle X_1 \times \cdots \times X_n, U, f \rangle$ with $f(\langle x_1, \ldots, x_n \rangle, u) = \langle f_1(x_1, u), \ldots, f_n(x_n, u) \rangle$.

Consider an input sequence $\mathbf{u} = (u_t)_{t \geq 1}$ that is η-convergent in U. For each $i \in [1..n]$, let $x^0_i \in X_i$. For each $i \in [1..n]$, we define the state sequence $(x_i, t)_{t \geq 1}$ with $x_{i,t} = D_i(x^0_i, u_1 \cdots u_t)$ the state of D_n on input $u_1 \cdots u_t$ when started from state x^0_i; and we define the state sequence $(x_t)_{t \geq 1}$ with $x_t = D(\langle x^0_1, \ldots, x^0_n \rangle, u_1 \cdots u_t)$ the state of D on input $u_1 \cdots u_t$ when started from state $\langle x^0_1, \ldots, x^0_n \rangle$.

By the form of the dynamics function of D noted above, we have that the state sequence $(x_t)_{t \geq 1}$ of the given composition D equals the state sequence $(\langle x_{1,t}, \ldots, x_{n,t} \rangle)_{t \geq 1}$.

Now, for every $i \in [1..n]$, we have that the sequence $(x_{i,t})_{t \geq 1}$ is η-convergent since $(u_t)_{t \geq 1}$ is η-convergent by assumption, and D_i is aperiodic by assumption. Therefore, the sequence $(\langle x_{1,t}, \ldots, x_{n,t} \rangle)_{t \geq 1}$ is η-convergent, and hence the state sequence $(x_t)_{t \geq 1}$ is η-convergent, showing that D is aperiodic. □

Lemma 23. *Let $C = D_1 \ltimes \cdots \ltimes D_n$ be a cascade of η-finite aperiodic dynamics. Then C is aperiodic.*

Proof. We proceed by induction on n. In the base case $n = 1$ and the claim holds by assumption. In the inductive case $n > 1$ and we assume by induction that the cascade $C_{n-1} = D_1 \ltimes \cdots \ltimes D_{n-1}$ is aperiodic. We have that C_{n-1} and D_n are of the form

$$C_{n-1} = \langle X_{n-1}, U, f_{n-1} \rangle, \quad D_n = \langle X_n, U \times X_{n-1}, f_n \rangle.$$

Then, we have that C is of the form

$$C = \langle X_{n-1} \times X_n, U, f \rangle, \qquad f(\langle x_{n-1}, x_n \rangle, u) = \langle f_{n-1}(x_{n-1}, u), f_n(x_n, \langle u, x_{n-1} \rangle) \rangle$$

Consider an input sequence $\mathbf{u} = (u_t)_{t \geq 1}$ that is η-convergent in U. Let $x^0_{n-1} \in X_{n-1}$ and let $x^0_n \in X_n$. We define the state sequence $(x_{n-1,t})_{t \geq 1}$ with $x_{n-1,t} = C_{n-1}(x^0_{n-1}, u_1 \cdots u_t)$

the state of C_{n-1} on input $u_1 \cdots u_t$ when started from state x^0_{n-1}. Then we define the state sequence $(x_{n,t})_{t\geq 1}$ with $x_{n,t} = D_n(x^0_n, \langle u_1, x_{n-1,1}\rangle \cdots \langle u_t, x_{n-1,t-1}\rangle)$ the state of D_n on input $\langle u_1, x_{n-1,1}\rangle \cdots \langle u_t, x_{n-1,t-1}\rangle$ when started from state x^0_n. Finally, we define the state sequence $(x_t)_{t\geq 1}$ with $x_t = C(\langle x^0_{n-1}, x^0_n\rangle, u_1 \cdots u_t)$ the state of C on input $u_1 \cdots u_t$ when started from state $\langle x^0_{n-1}, x^0_n\rangle$.

By the form of the dynamics function of C noted above, we have that the state sequence $(x_t)_{t\geq 1}$ of the given cascade C equals the state sequence $(\langle x_{n-1,t}, x_{n,t}\rangle)_{t\geq 1}$.

Now, we have that the sequence $(x_{n,t})_{t\geq 1}$ is η-convergent since $(u_t)_{t\geq 1}$ is η-convergent by assumption, and C_{n-1} is aperiodic by the inductive hypothesis. In turn, this implies that the state sequence $(x_{n,t})_{t\geq 1}$ is η-convergent, since D_n is aperiodic by assumption. Therefore, the sequence $(\langle x_{n-1,t}, x_{n,t}\rangle)_{t\geq 1}$ is η-convergent, and hence the state sequence $(x_t)_{t\geq 1}$ is η-convergent, showing that C is aperiodic. □

Lemma 24. *Every finite MinMax Recurrent Unit of state dimension one is aperiodic.*

Proof. Let $D = \langle X, U, f \mid R, s\rangle$ be a finite MinMax Recurrent Unit of state dimension one. Let $(u_t)_{t\geq 1}$ be an η-convergent sequence in U. Since U is finite, its η-components are singleton, and hence all inputs u_t are the same input $u \in U$. Let $x \in X$, and let $\tau(x) := f(x, u)$. We have

$$\begin{aligned}
\tau(\tau(x)) &= f(f(x,u),u) \\
&= \Big(R(u) \odot \big((R(u) \odot x) \oplus s(u)\big)\Big) \oplus s(u) \\
&= \big(R(u) \odot R(u) \odot x) \oplus (R(u) \odot s(u))\big) \oplus s(u) \\
&= (R(u) \odot x) \oplus (R(u) \odot s(u)) \oplus s(u) \\
&= \big(R(u) \odot (x \oplus s(u))\big) \oplus s(u)
\end{aligned}$$

and hence

$$\begin{aligned}
\tau(\tau(\tau(x))) &= R(u) \odot \Big(\big(R(u) \odot (x \oplus s(u))\big) \oplus s(u)\Big) \oplus s(u) \\
&= \Big(\big(R(u) \odot R(u) \odot (x \oplus s(u))\big) \oplus R(u) \odot s(u)\Big) \oplus s(u) \\
&= \Big(\big(R(u) \odot (x \oplus s(u))\big) \oplus R(u) \odot s(u)\Big) \oplus s(u) \\
&= \big(R(u) \odot (x \oplus s(u) \oplus s(u))\big) \oplus s(u) \\
&= \big(R(u) \odot (x \oplus s(u))\big) \oplus s(u) \\
&= \tau(\tau(x)).
\end{aligned}$$

It follows that $\tau^n = \tau^{n+1}$ for all $n \geq 2$.

Let us consider the state sequence $(x_t)_{t\geq 1}$ with $x_t = D(x, u_1 \cdots u_t)$ the state of D on input $u_1 \cdots u_t$ when started from state x. Since every u_t equals u, we have that $x_t = \tau^t(x)$. Since $\tau^t = \tau^{t+1}$ for all $t \geq 2$, we have that $x_t = \tau^2(x)$ for all $t \geq 2$, and hence the sequence $(x_t)_{t\geq 1}$ is η-convergent, showing that D is aperiodic. □

The following theorem corresponds to the third claim of Theorem 1.

Theorem 28. *Let S be a MinMax RNC of state degree one. Every regular function F that is not group-free cannot be implemented by S.*

Proof. Let F be a function implemented by a MinMax Recurrent Cascade of state degree one. By Lemma 21, we have that F can be implemented by a MinMax Recurrent Cascade S having state degree one, and a finite input and state space, and hence it is trivially η-finite. Then, by Lemma 24, we have that each MinMax Recurrent Unit of S is aperiodic, and hence the dynamics of S are aperiodic by Lemmas 22 and 23. By Theorem 16, it follows that $\mathcal{C}_\eta(S)$ is group-free. Then F is group-free since, by Theorem 13 we have that $\mathcal{C}_\eta(S)$ can implement F. The proof applies to languages as well, since recognising a language means implementing its characteristic function. □

Theorem 1. *Every regular function of degree n can be implemented by an η-finite MinMax RNC of state degree n. Every function that is not regular cannot be implemented by a MinMax RNC. Every function that is not group-free cannot be implemented by a MinMax RNC of state degree one.*

Proof. The three claims of the theorem correspond to Theorems 26, 27, and 28 respectively. □

Next we provide the floating-point counter-part of the theorem above.

Theorem 29. *Every regular function can be implemented by a floating-point MinMax RNC of state degree equal to the degree of the function. Any function that is not regular cannot be implemented by a floating-point MinMax RNC. Any function that is not group-free cannot be implemented by a floating-point MinMax RNC of state degree one.*

Proof. The three claims of the theorem correspond to Theorems 26, 27, and 28 respectively. □

Corollary 19. *The functions that can be implemented by floating-point MinMax RNCs are the regular functions, and the group-free functions if restricted to state degree one. The languages that can be recognised by floating-point MinMax RNCs are the regular languages, and the star-free languages if restricted to state degree one.*

I.5 Positive-margin expressivity of MinMax RNCs

Theorem 30 (Positive-margin expressivity of MinMax RNCs)**.** *The positive-margin expressivity of MinMax RNCs is exactly the class of regular languages.* End of Theorem 30.

Proof. We show the two inclusions separately.

By Theorem 1, every regular language can be recognised by an η-finite MinMax RNC. Hence, by Corollary 13, the positive-margin expressivity of MinMax RNCs includes the class of regular languages.

Conversely, recognition with positive margin is, in particular, recognition. Therefore, the positive-margin expressivity of MinMax RNCs is contained in their ordinary formal-language expressivity. Then the second inclusion follows by Theorem 1. □

J Algorithms for MinMax RNCs

We present the algorithms to evaluate MinMax RNCs—their pseudocodes follow right below. For sequential evaluation, we simply follows the recurrence. For parallel evaluation, we first note that any two MinMax Recurrence operators, given by their matrix vector pairs $\langle A_1, b_1\rangle, \langle A_2, b_2\rangle$, can be composed via the following associative operator:

$$\langle A_1, b_1\rangle \circledast \langle A_2, b_2\rangle := \langle A_2 \otimes A_1, (A_2 \otimes b_1) \oplus b_2\rangle.$$

Then we apply parallel scan [31, 41].

Thus, the algorithms in this appendix are mainly a basis for building rigorous proofs of the theorems in Appendix K, where we show correctness and complexity of all the present algorithms, and overall of MinMax Recurrence and MinMax RNCs.

J.1 Preliminary remarks on our Pseudocode

We use common instructions with a notation that should be self-clear.

Parallel for. To describe parallel algorithms, we use the *parallel for* instruction, cf. [41].

in parallel for $\mathtt{i} \leftarrow e_1$ **to** e_2 **do** *instructions*

It corresponds to executing the block *instructions* in parallel providing to each thread a different index $\mathtt{i}$ ranging from the value of expression e_1 to the value of expression e_2. Furthermore, we avoid concurrent writes, thus satisfying the *Concurrent Read Exclusive Write (CREW)* constraint.

0-indexed tensors. In this appendix and in Appendix K we adopt 0-indexed real-valued tensors. We use $\mathtt{shape}(A)$ to obtain the shape of a tensor. We use $A \parallel B$ to concatenate tensors only when they are vectors, making its meaning straightforward.

No side effects. Although we adopt an imperative pseudocode with side effects, we avoid to assign twice any variable or entry of a tensor. This makes the correctness proofs easier to develop.

J.2 Auxiliary Algorithms

In this subsection we provide the pseudocode of the auxiliary algorithms for parallel evaluation.

- Algorithm 1 is an immediate application of the parallel sum algorithm (cf. [41]) by replacing $+$ with $\odot$, since the algorithm works for all associative operators, including $\odot$.
- Algorithm 2 is analogous to the above Algorithm 2, except that now $+$ is replaced with $\oplus$.
- Algorithm 3 is a straightforward computation entry-wise.
- Algorithm 4 is a straightforward algorithm for parallel matrix multiplication, with arithmetic addition and multiplication replaced by $\{\oplus, \odot\}$.
- Algorithm 5 uses the above algorithms to evaluate the $\circledast$ operator, to compose a pair of MinMax recurrences, used by parallel scan to compute multi-step compositions.
- Algorithm 6 applies the MinMax Recurrence operator to a given vector (specifically, the given vector will be a state vector). It is used as the last step after calling parallel scan.

J.3 Sequential evaluation

We describe the sequential evaluation algorithm.

- SeqRecEval is the algorithm for sequential evaluation of MinMax Recurrence. It is a straightforward algorithm that computes in order following the recurrence.
- SeqLayerEval is the algorithm for sequential evaluation of a MinMax Recurrent Layer. It evaluates the neurons' input functions by assuming a subroutine to evaluate MLPs, and by calling SeqRecEval on the result of their their evaluation. Finally, it evaluates the layer's output function, again by calling the subroutine for MLPs.
- SeqCascEval is the algorithm for sequential evaluation of a MinMax RNC. It simply evaluates layers iteratively by calling SeqRecEval.

J.4 Parallel Evaluation

We present the parallel evaluation a algorithm.

J.4.1 MinMax Parallel Scan

Parallel scan [31, 39, 40] is the core of the parallel evaluation algorithm.

Here we define the operator we use to compose MinMax Recurrence operators.

Definition 211. A *MinMax-recurrence pair* over $V \subseteq \mathbb{R}$ of dimension $n \in \mathbb{N}^{>0}$ is a pair $M = \langle A, b\rangle$ with $A \in V^{n\times n}$, $b \in V^n$, and its *MinMax-recurrence map* is $\mathcal{A}_M(x) := (A \otimes x) \oplus b$. MinMax-recurrence pairs $M_1 = \langle A_1, b_1\rangle$ and $M_2 = \langle A_2, b_2\rangle$ are *composable* if they have the same dimension; in this case, their *MinMax-recurrence composition* is $M_1 \circledast M_2 := \langle A_2 \otimes A_1, (A_2 \otimes b_1) \oplus b_2\rangle$.

Then, parallel scan is used to evaluate all (exclusive) prefixes of an expression of the form:

$$\langle A_1, b_1\rangle \circledast \cdots \circledast \langle A_t, b_t\rangle. \tag{43}$$

The affine MinMax operator $\circledast$ is associative and it admits an identity element, which is the reason why we can use parallel scan—formally proved in Appendix K.

First, parallel scan requires the expression in (43) to be a monoidal expression. Namely, the operator $\circledast$ must be associative, and there must be an identity element, which is[9] an element $\langle E, e\rangle$ such that

$$\langle E, e\rangle \circledast \langle A, b\rangle = \langle A, b\rangle \circledast \langle E, e\rangle = \langle A, b\rangle$$

for all elements $\langle A, b\rangle$ that can be generated by any intermediate composition

$$\langle A, b\rangle = \langle A_{t_1}, b_{t_1}\rangle \circledast \cdots \circledast \langle A_{t_2}, b_{t_2}\rangle.$$

Algorithms. We describe parallel scan and its subroutine for the identity element.

- Algorithm 7 generates the identity element $\langle E, e\rangle$ out of two given values $v_{\min}$ and $v_{\max}$, which in the overall algorithm will be the largest and smallest scalars returned by unit's input function on its entire input sequence, also including the entries of the initial state. Intuitively, $(v_{\min}, v_{\max})$ will mimick $(-\infty, +\infty)$ which act as $(0, 1)$ in the MinMax semiring $(\oplus, \odot, -\infty, +\infty)$.[10] Thus, E is similar to the identity matrix $I_{n,n}$ (replacing 1's with $+\infty$ and 0's with $-\infty$) and e is similar to the null vector 0_n (replacing 0's with $-\infty$).
- Algorithm 8 follows the standard parallel scan algorithm (e.g., see [41]), combining pairs of elements of the given expression, with some careful handling of the odd and even cases. It calls Algorithm 7 in the case case, as it computes exclusive prefixes, hence identity on a given expression of length 1.

J.4.2 Parallel Evaluation

Parallel evaluation of MinMax Recurrence, and then of layers and cascades, is straightforward once parallel scan computes prefixes of recurrences.

- ParallelRecEval is the algorithm for parallel evaluation of MinMax Recurrence. It computes $v_{\min}, v_{\max}$ that are required by parallel scan. Then, it calls parallel scan (Algorithm 8) to compute the sequence of matrices corresponding to prefixes of compositions of MinMax Recurrence. Then, in parallel for every sequence step t, it includes the recurrence step at time t (since parallel scan returns exclusive prefixes), and finally it applies the overall composition to the initial state, hence obtaining the state at time t.
- ParLayerEval is the algorithm for parallel evaluation of a MinMax Recurrent Layer. It is as SeqLayerEval, except that it handles each neuron in parallel, it calls ParallelRecEval, and then computes the output's layer in parallel wrt sequence steps.
- ParCascEval is the algorithm for parallel evaluation of a MinMax RNC. It is as ParCascEval, except that it calls ParLayerEval to evaluate layers.

[9] allowing ourselves not to be entirely precise in this informal explanation for the sake for providing the intuition

[10] Potentially we could have used the MinMax semiring $(\oplus, \odot, -\infty, +\infty)$ throughout the paper, instead of using only the MinMax algebra $(\oplus, \odot)$. However, we have preferred not to introduce infinities.

Algorithm 1: Minimum element of an array computed in parallel

Input : Array A of length $n \geq 1$
Output : $\bigodot_{i=0}^{n-1} A[i]$

```
Function ParallelMin(A):
    if n = 1 then
        return A[0];
    else
        m ← ⌊n/2⌋;
        allocate array B of length m;
        in parallel for i ∈ [0..m − 2] do
            B[i] ← A[2i] ⊙ A[2i + 1];
        if n is even then
            B[m − 1] ← A[n − 2] ⊙ A[n − 1];
        else
            B[m − 1] ← A[n − 3] ⊙ A[n − 2] ⊙ A[n − 1];
        return ParallelMin(B);
```

Algorithm 2: Maximum element of an array computed in parallel

Input : Array A of length n
Output : $\bigoplus_{i=0}^{n-1} A[i]$

```
Function ParallelMax(A):
    if n = 1 then
        return A[0];
    else
        m ← ⌊n/2⌋;
        allocate array B of length m;
        in parallel for i ∈ [0..m − 2] do
            B[i] ← A[2i] ⊕ A[2i + 1];
        if n is even then
            B[m − 1] ← A[n − 2] ⊕ A[n − 1];
        else
            B[m − 1] ← A[n − 3] ⊕ A[n − 2] ⊕ A[n − 1];
        return ParallelMax(B);
```

Algorithm 3: Parallel matrix addition

Input : Matrices $A, B \in \mathbb{R}^{m \times n}$
Output : Matrix $C \in \mathbb{R}^{m \times n}$ such that $C = A \oplus B$

Function `ParallelAdd`(A, B)**:**
    allocate matrix C of shape (m, n);
    in parallel for $i \leftarrow 0$ **to** $m - 1$ **do**
        in parallel for $j \leftarrow 0$ **to** $n - 1$ **do**
            $C[i, j] \leftarrow A[i, j] \oplus B[i, j]$;
    return C;

Algorithm 4: Parallel matrix multiplication

Input : Matrix $A \in \mathbb{R}^{m \times p}$, matrix $B \in \mathbb{R}^{p \times n}$
Output : Matrix $C \in \mathbb{R}^{m \times n}$ such that $C = A \otimes B$

Function `ParallelMult`(A, B)**:**
    allocate matrix C of shape (m, n);
    allocate tensor T of shape (m, n, p);
    in parallel for $i \leftarrow 0$ **to** $m - 1$ **do**
        in parallel for $j \leftarrow 0$ **to** $n - 1$ **do**
            in parallel for $k \leftarrow 0$ **to** $p - 1$ **do**
                $T[i, j, k] \leftarrow A[i, k] \odot B[k, j]$;
            $C[i, j] \leftarrow$ `ParallelMax`$(T[i, j])$;
    return C;

Algorithm 5: Parallel MinMax Composition

Input : $(A_1, B_1) \in \mathbb{R}^{n \times n} \times \mathbb{R}^n$, $(A_2, B_2) \in \mathbb{R}^{n \times n} \times \mathbb{R}^n$
Output : $(A, B) \in \mathbb{R}^{n \times n} \times \mathbb{R}^n$ such that $\langle A, B \rangle = \langle A_1, B_1 \rangle \circledast \langle A_2, B_2 \rangle$

Function `ParallelComp`(A_1, B_1, A_2, B_2)**:**
    allocate tensor A of shape (n, n);
    allocate arrays B, Z of length n;
    $A \leftarrow$ `ParallelMult`(A_2, A_1);
    $Z \leftarrow$ `ParallelMult`(A_2, B_1);
    $B \leftarrow$ `ParallelAdd`(Z, B_2);
    return (A, B);

Algorithm 6: Parallel MinMax Operator Application

Input : tensor A of shape (N, N), array B of length N, array X of length N
Output : tensor Y such that $Y = (A \otimes X) \oplus B$

Function `ParallelApply`(A, B, X)**:**
    allocate arrays Z, Y of length N;
    $Z \leftarrow$ `ParallelMult`(A, X);
    $Y \leftarrow$ `ParallelAdd`(Z, B);
    return Y;

Algorithm 7: Compute identity element in parallel

Input : scalars $v_{\min}, v_{\max} \in \mathbb{R}$ and $n \in \mathbb{N}^{>0}$

Output : identity (E, e) for MinMax monoid $\mathcal{M}_{[v_{\min}, v_{\max}], n}$

Function `ParallelIdentity`$(v_{\min}, v_{\max}, n)$**:**
- **allocate** tensor E of shape (N, N);
- **in parallel for** $i \in [0..N-1)$ **do**
  - **in parallel for** $j \in [0..N)$ **do**
    - **if** $i = j$ **then** $E[i, j] \leftarrow v_{\max}$;
    - **else** $E[i, j] \leftarrow v_{\min}$;
- **allocate** array e of length N;
- **in parallel for** $i \in [0..N)$ **do**
  - $e[i] \leftarrow v_{\min}$;
- **return** (E, e);

Algorithm 8: Parallel Scan

Input : tensor A of shape (T, N, N), tensor B of shape (T, N), scalars $v_{\min}, v_{\max}$;

Output : tensors (C, D) with $(C[t], D[t]) = (A[0], B[0]) \circledast \cdots \circledast (A[t-1], B[t-1]) \circledast \mathcal{E}$ for all $t \in [0..T)$, with $\mathcal{E}$ the identity of the MinMax monoid $\mathcal{M}_{[v_{\min}, v_{\max}], N}$

Function `ParallelScan`$(A, B, v_{\min}, v_{\max})$**:**
- $(T, N) \leftarrow$ `shape`(B);
- **if** $T = 1$ **then**
  - **return** `ParallelIdentity`$(v_{\min}, v_{\max}, N)$;
- $m \leftarrow \lfloor T/2 \rfloor$; $\ell \leftarrow \lceil T/2 \rceil$;
- **allocate** tensor X of shape (ℓ, N, N);
- **allocate** tensor Y of shape (ℓ, N);
- **in parallel for** $t \in [0..m-1]$ **do**
  - $(X[t], Y[t]) \leftarrow$ `ParallelComp`$(A[2t], B[2t], A[2t+1], B[2t+1])$;
- **if** T *is odd* **then**
  - $(X[\ell-1], Y[\ell-1]) \leftarrow (A[T-1], B[T-1])$;
- $(Z, W) \leftarrow$ `ParallelScan`$(X, Y, v_{\min}, v_{\max})$;
- **allocate** tensor C of shape (T, N, N);
- **allocate** tensor D of shape (T, N);
- **in parallel for** $t \in [0..T-1]$ **do**
  - **if** t *is even* **then**
    - $(C[t], D[t]) \leftarrow (Z[t/2], W[t/2])$;
  - **else**
    - $(C[t], D[t]) \leftarrow$ `ParallelComp`$(Z[(t-1)/2], W[(t-1)/2], A[t-1], B[t-1])$;
- **return** (C, D);

Function `SeqRecEval`(X_{init}, A, B)

Input : tensors $\langle X_{\text{init}}, A, B\rangle$ with $X_{\text{init}} \in \mathbb{R}^N$, $A \in \mathbb{R}^{T\times N\times N}$, $B \in \mathbb{R}^{T\times N}$;

Output : tensor X with $X[0] = \circledast(X_{\text{init}}, A[0], B[0])$ and $X[t] = \circledast(X[t-1], A[t], B[t])$ for all $t \in [1..T)$;

$(T, N) \leftarrow \text{shape}(B)$;
allocate tensor X of shape (T, N);
allocate tensor Z of shape $(T+1, N+1, N)$;
$Z[0, N] \leftarrow X_{\text{init}}$;
for $t \leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    $Z[t, 0, i-1] \leftarrow B[t-1, i-1]$;
    for $j \leftarrow 1$ **to** N **do**
      $Z[t, j, i-1] \leftarrow Z[t, j-1, i-1] \oplus \big(A[t-1, i-1, j-1] \odot Z[t-1, N, j-1]\big)$;
  $X[t-1] \leftarrow Z[t, N]$;
return X;

Function `SeqLayerEval`(W, U)

Input : (W, U) where W are the weights of a MinMax Layer S and $U \in \mathbb{R}^{T\times d_{\text{in}}}$ for d_{in} the input dimension of S;

Output : $Y \in \mathbb{R}^{T\times d_{\text{out}}}$ for d_{out} the output dimension of S, such that $Y = S(U)$;

$(T, d_{\text{in}}) \leftarrow \mathsf{shape}(U)$; $(n_{\text{units}}, d_{\text{state}}) \leftarrow \mathsf{shape}(X^{\text{init}})$; $(d_{\text{out}}, _) \leftarrow \mathsf{shape}(W^{\text{out}})$;
$(W^{\text{R}}, W^{\text{s}}, W^{\text{out}}, X^{\text{init}}) \leftarrow W$;
allocate tensor X of shape $(T+1,\ n_{\text{units}} \cdot d_{\text{state}})$;
for $i \leftarrow 0$ **to** $n_{\text{units}} - 1$ **do**
  allocate tensor A_i of shape $(T, d_{\text{state}}, d_{\text{state}})$;
  allocate tensor B_i of shape (T, d_{state});
  for $t \leftarrow 0$ **to** $T - 1$ **do**
    $A_i[t] \leftarrow \texttt{EvaluateMLP}(W^{\text{R}}[i], U[t])$;
    $B_i[t] \leftarrow \texttt{EvaluateMLP}(W^{\text{s}}[i], U[t])$;
  $p_i \leftarrow i \cdot d_{\text{state}}$;
  $q_i \leftarrow (i+1) \cdot d_{\text{state}}$;
  $X[0,\ p_i : q_i] \leftarrow X^{\text{init}}[i]$;
  $X[1:,\ p_i : q_i] \leftarrow \texttt{SeqRecEval}(X^{\text{init}}[i], A_i, B_i)$;
allocate tensor Y of shape (T, d_{out});
for $t \leftarrow 0$ **to** $T - 1$ **do**
  $Y[t] \leftarrow \texttt{EvaluateMLP}(W^{\text{out}},\ X[t] \parallel X[t+1] \parallel U[t])$;
return Y;

Function `SeqCascEval`(W, U)

Input : $\langle W, U \rangle$ where W are the weights of an RNC S, and $U \in \mathbb{R}^{T \times d_{\text{in}}}$ for d_{in} the input dimension of S;

Output : $Y \in \mathbb{R}^{T \times d_{\text{out}}}$ for d_{out} the output dimension of S, such that $Y = S(U)$;

$n_{\text{layers}} \leftarrow \mathsf{length}(W)$;
$Y \leftarrow U$;
for $i \leftarrow 0$ **to** $n_{\text{layers}} - 1$ **do**
    $Y \leftarrow$ `SeqLayerEval`$(W[i], Y)$;
return Y;

Function `ParallelRecEval`(X_{init}, A, B)

Input : array X_{init} of length N, tensor A of shape (T, N, N), tensor B of shape (T, N);

Output : tensor X of shape (T, N) such that $X[0] = \circledast(X_{\text{init}}, A[0], B[0])$ and $X[t] = \circledast(X[t-1], A[t], B[t])$ for every $t \in (0..T)$;

$(T, N) \leftarrow \mathsf{shape}(B)$;
$v_{\min} \leftarrow$ `ParallelMin`$(X_{\text{init}} \parallel \text{flat}(A) \parallel \text{flat}(B))$;
$v_{\max} \leftarrow$ `ParallelMax`$(X_{\text{init}} \parallel \text{flat}(A) \parallel \text{flat}(B))$;
$(C, D) \leftarrow$ `ParallelScan`$(A, B, v_{\min}, v_{\max})$;
allocate tensors X, W of shape (T, N);
allocate tensor Z of shape (T, N, N);
in parallel for $t \leftarrow 0$ **to** $T - 1$ **do**
    $(Z[t], W[t]) \leftarrow$ `ParallelComp`$(C[t], D[t], A[t], B[t])$;
    $X[t] \leftarrow$ `ParallelApply`$(Z[t], W[t], X_{\text{init}})$;
return X;

Function `ParLayerEval(`W, U`)`

Input : (W, U) where W are the weights of a MinMax Layer S and $U \in \mathbb{R}^{T \times d_{\text{in}}}$ for d_{in} the input dimension of S;

Output : Y such that $Y = S(U)$;

$(T, d_{\text{in}}) \leftarrow \mathsf{shape}(U)$; $(n_{\text{units}}, d_{\text{state}}) \leftarrow \mathsf{shape}(X^{\text{init}})$; $(d_{\text{out}}, _) \leftarrow \mathsf{shape}(W^{\text{out}})$;
$(W^{\text{R}}, W^{\text{s}}, W^{\text{out}}, X^{\text{init}}) \leftarrow W$;
allocate tensor X of shape $(T + 1,\ n_{\text{units}} \cdot d_{\text{state}})$;
in parallel for $i \leftarrow 0$ **to** $n_{\text{units}} - 1$ **do**
  allocate tensor A_i of shape $(T, d_{\text{state}}, d_{\text{state}})$;
  allocate tensor B_i of shape (T, d_{state});
  in parallel for $t \leftarrow 0$ **to** $T - 1$ **do**
    $A_i[t] \leftarrow \texttt{EvaluateMLP}(W^{\text{R}}[i], U[t])$;
    $B_i[t] \leftarrow \texttt{EvaluateMLP}(W^{\text{s}}[i], U[t])$;
  $p_i \leftarrow i \cdot d_{\text{state}}$;
  $q_i \leftarrow (i + 1) \cdot d_{\text{state}}$;
  $X[0,\ p_i : q_i] \leftarrow X^{\text{init}}[i]$;
  $X[1 :,\ p_i : q_i] \leftarrow \texttt{ParallelRecEval}(X^{\text{init}}[i], A_i, B_i)$;
allocate tensor Y of shape (T, d_{out});
in parallel for $t \leftarrow 0$ **to** $T - 1$ **do**
  $Y[t] \leftarrow \texttt{EvaluateMLP}(W^{\text{out}},\ X[t] \parallel X[t + 1] \parallel U[t])$;
return Y;

Function `ParCascEval(`W, U`)`

Input : $\langle W, U \rangle$ where W are the weights of an RNC S, and $U \in \mathbb{R}^{T \times d_{\text{in}}}$ for d_{in} the input dimension of S;

Output : $Y \in \mathbb{R}^{T \times d_{\text{out}}}$ for d_{out} the output dimension of S, such that $Y = S(U)$;

$n_{\text{layers}} \leftarrow \mathsf{length}(W)$;
$Y \leftarrow U$;
for $i \leftarrow 0$ **to** $n_{\text{layers}} - 1$ **do**
  $Y \leftarrow \texttt{ParLayerEval}(W[i], Y)$;
return Y;

K Computational Complexity of MinMax RNCs

In this section we prove the theorems of Section 5. The theorems are proved in Section K.3, after discussing the setup in which the results are proved (Section K.1), and after formally defining the relevant computation problems (Section K.2).

K.1 Model of computation

The complexity results are established considering the following models of computation.

Sequential case. The sequential complexity results are proven under the assumptions of *Random-Access Machine (RAM)* on real-valued arithmetic, cf. [61]. A RAM has a finite number of registers and, in one computation step, it may perform one of the following operations:

- write to a register the scalar value $a \circ b$, where a, b are scalar values read from some registers, for any operation $\circ \in \{+, -, \times, \div, \odot, \oplus\}$ and $b \neq 0$ in the case of division;
- write to a register the scalar value $\lceil a \rceil$ or $\lfloor a \rfloor$ for a a scalar value read from a register;
- compare the value of two registers and branch accordingly.

The instructions in the pseudocode we use are in line with RAM, while providing a higher-level specification language (e.g., registers correspond to variables and entries of a tensor, branching is involved in if-then-else and in for loops). Importantly, in our pseudocode we only use the elementary arithmetic operations mentioned above, whose cost is assumed to be 1.

Parallel case. The parallel complexity results are proven considering *Parallel RAM on real-valued arithmetic with Concurrent Read and Exclusive Write (PRAM-CREW)*, cf. [41]. A PRAM-CREW has an arbitrary number of processors that carry out the computation proceeding by rounds. In each round, the processors simultaneously perform one of the operations described above for the RAM model, under the condition that two processors never write to the same register in the same round. The *work* is the total number of operations performed by the processors, and the *depth* is the number of rounds. The instructions in our pseudocode are in line with the PRAM-CREW, while providing a higher-level specification language. Compared to the sequential case, we now have the 'parallel for' instruction, which allows us to describe simultaneous computations carried out by the processors. Brent's Theorem translates work and depth to execution time on PRAM-CREW.

Theorem 31 (Brent's Theorem, cf. Theorem 25.1 in [41])**.** *Any algorithm having work W and depth D can be executed in $O(W/P + D)$ steps in the CREW-PRAM model.*

K.2 Computation problems

We define the relevant compuation problems, formalising the tasks for which we want to show their complexity.

K.2.1 Background

Definition 212. A *computation problem over* $\mathbb{R}$ is a triple $\langle I, O, f \rangle$ where I is a set of tuples of tensors called *valid inputs* or simply *inputs*, O is a set of tuples of tensors called *possible outputs* or simply *outputs*, $f : I \to O$ is a function that maps every input element $x \in I$ to an output element $f(x) \in O$ called *the expected output for input* x. The *size of an input* $x \in I$ is the number of scalar values in the tensors of x.

Definition 213. A program $\mathcal{P}$ *solves* a computation problem $\langle I, O, f, s \rangle$ if, for every input $x \in I$, we have that x is a valid input for program $\mathcal{P}$ and the output $\mathcal{P}(x)$ of the program equals $f(x)$.

Definition 214. For $T : \mathbb{N} \to \mathbb{N}$ a monotonic function, *a sequential program $\mathcal{P}$ solves a computation problem $\langle I, O, f \rangle$ in time T* if, for every input $x \in I$, we have that x is a valid input for $\mathcal{P}$, the the output $\mathcal{P}(x)$ of the program equals $f(x)$, and the execution time of $\mathcal{P}$ on input x is at most $T(n)$ for n the size of input x.

Definition 215. For $W, D : \mathbb{N} \to \mathbb{N}$ monotonic functions, *a parallel program $\mathcal{P}$ solves a computation problem $\langle I, O, f \rangle$ with depth D and work W* if, for every input $x \in I$, we have that x is a valid input

for $\mathcal{P}$, the output of the program $\mathcal{P}(x)$ equals $f(x)$, the depth of the execution of $\mathcal{P}$ on x is at most $D(n)$ for n the size x, and the work of the execution of $\mathcal{P}$ on x is at most $W(n)$.

Definition 216. For $T, P : \mathbb{N} \to \mathbb{N}$ monotonic functions, *a parallel program $\mathcal{P}$ solves a computation problem $\langle I, O, f\rangle$ in time T having P processors available* if, for every input $x \in I$, we have that x is a valid input for $\mathcal{P}$, the output of the program $\mathcal{P}(x)$ equals $f(x)$, and the execution time of program $\mathcal{P}$ on input x having $P(n)$ processors available is at most $T(n)$ for n is the size of input x.

Definition 217. We say that a computation problem can be solved sequentially in time T if there exists a sequential program that solves it in time T. We say that a computation problem can be solved in parallel with work W and depth D if there exists a parallel program that solves it with work W and depth D. We say that a computation problem can be solved in parallel in time T having P processors available if there exists a parallel program that solves it in time T having P processors available.

K.2.2 Definition of the considered computation problems

Definition 218 (MinMax Recurrence Evaluation)**.** *MinMax Recurrence Evaluation* is the computation problem where inputs and outputs are respectively given by

$$I := \bigcup_{T=1}^{\infty} \bigcup_{N=1}^{\infty} I_{T,N} \quad \text{with } I_{T,N} := \big\{\langle x_{\text{init}}, A, b\rangle \mid x_{\text{init}} \in \mathbb{R}^N,\ A \in \mathbb{R}^{T\times N\times N},\ b \in \mathbb{R}^{T\times N}\big\},$$

$$O := \bigcup_{T=1}^{\infty} \bigcup_{N=1}^{\infty} O_{T,N} \quad \text{with } O_{T,N} := \big\{X \mid X \in \mathbb{R}^{T\times N}\big\},$$

and the expected output for input $\langle x_{\text{init}}, A, b\rangle \in I_{T,N}$ is $X \in O_{T,N}$ satisfying

$$X[0] = \circledast\big(x_{\text{init}}, A[0], b[0]\big), \quad X[t] = \circledast\big(X[t-1], A[t], b[t]\big) \quad \forall t \in [1..T).$$

For inputs in $I_{T,N}$, we call T the *length* or *number of steps*, and we call N the *state dimension*.

End of Definition 218.

Definition 219 (Representation of MinMax RNCs)**.** The size of a linear function is $\mathbf{s} = \langle d_{\text{in}}, d_{\text{out}}\rangle$. The weights for a linear function of size $\mathbf{s}$ are

$$\mathbf{W}^{\text{lin}}_{\mathbf{s}} := \{\langle A, b\rangle \mid A \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}},\ b \in \mathbb{R}^{d_{\text{out}}}\}$$

The linear function having weights $W = \langle A, b\rangle \in \mathbf{W}^{\text{linear}}_{\mathbf{s}}$ is the function $f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ given by

$$f(x) = Ax + b,$$

and denoted by $\text{Lin}(W) := f$.

The size of an MLP is $\mathbf{s} = \langle d_{\text{in}}, d_{\text{out}}, d_{\text{h}}, n_{\text{h}}\rangle$ with input dimension d_{in}, output dimension d_{out}, hidden dimension d_{h}, and number of hidden layers n_{h}. The weights for an MLP of size $\mathbf{s}$ are

$$\mathbf{W}^{\text{mlp}}_{\mathbf{s}} := \{\langle W^{\text{in}}, \langle W^{\text{h}}_0, \ldots, W^{\text{h}}_{n_{\text{h}}-1}\rangle, W^{\text{out}}\rangle \mid W^{\text{in}} \in \mathbf{W}^{\text{lin}}_{d_{\text{in}},d_{\text{h}}},\ W^{\text{h}}_i \in \mathbf{W}^{\text{lin}}_{d_{\text{h}},d_{\text{h}}},\ W^{\text{out}} \in \mathbf{W}^{\text{lin}}_{d_{\text{h}},d_{\text{out}}}\}$$

The MLP having weights $W = \langle W^{\text{in}}, W^{\text{h}}, W^{\text{out}}\rangle \in \mathbf{W}^{\text{mlp}}_{\mathbf{s}}$ is the function $\mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ given by

$$\text{MLP}(W) := \text{Lin}(W^{\text{in}}) \circ \phi \circ \text{Lin}(W^{\text{h}}[0]) \circ \phi \circ \cdots \circ \text{Lin}(W^{\text{h}}[n_{\text{h}}-1]) \circ \phi \circ \text{Lin}(W^{\text{out}}),$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is the activation function applied element-wise.

The size of a MinMax Neural Layer is $\mathbf{s} = \langle d_{\text{in}}, d_{\text{mlp}}, n_{\text{mlp}}, d_{\text{out}}, n_{\text{units}}, d_{\text{state}}\rangle$ with input dimension d_{in}, hidden dimension of MLPs d_{mlp}, number of hidden layers of MLPs n_{mlp}, output dimension d_{out}, number of units n_{units}, and state dimension of each unit d_{state}.

The weights of the input functions of a MinMax Neural Layer of size $\mathbf{s}$ are

$$\mathbf{W}^{\text{layer,R}}_{\mathbf{s}} := \{\langle W^{\text{R}}_0, \ldots, W^{\text{R}}_{n_{\text{units}}-1}\rangle \mid W^{\text{R}}_i \in \mathbf{W}^{\text{mlp}}_{\mathbf{s}_{\text{R}}}\} \quad \text{with } \mathbf{s}_{\text{R}} = \langle d_{\text{in}},\ d^2_{\text{state}},\ \lceil d_{\text{mlp}}/n_{\text{units}}\rceil,\ n_{\text{mlp}}\rangle,$$

$$\mathbf{W}^{\text{layer,s}}_{\mathbf{s}} := \{\langle W^{\text{s}}_0, \ldots, W^{\text{s}}_{n_{\text{units}}-1}\rangle \mid W^{\text{s}}_i \in \mathbf{W}^{\text{mlp}}_{\mathbf{s}_{\text{s}}}\} \quad \text{with } \mathbf{s}_{\text{s}} = \langle d_{\text{in}},\ d_{\text{state}},\ \lceil d_{\text{mlp}}/n_{\text{units}}\rceil,\ n_{\text{mlp}}\rangle,$$

and the weights of a MinMax Neural Layer of size $\mathbf{s}$ are

$$\begin{aligned}\mathbf{W}^{\text{layer}}_{\mathbf{s}} := \{\langle W^{\text{R}}, W^{\text{s}}, W^{\text{out}}, X^{\text{init}}\rangle \mid\ & W^{\text{R}} \in \mathbf{W}^{\text{layer,R}}_{\mathbf{s}},\\ & W^{\text{s}} \in \mathbf{W}^{\text{layer,s}}_{\mathbf{s}},\\ & W^{\text{out}} \in \mathbf{W}^{\text{mlp}}_{\mathbf{s}_{\text{out}}},\\ & X^{\text{init}} \in \mathbb{R}^{n_{\text{units}}\times d_{\text{state}}}\}\end{aligned}$$

where

$$\mathbf{s}_{\text{out}} = \langle d_{\text{state}} \cdot n_{\text{units}},\ d_{\text{out}},\ d_{\text{mlp}},\ n_{\text{mlp}} \rangle.$$

The MinMax Layer $\text{Layer}(W)$ with weights $W = \langle W^{\text{R}}, W^{\text{s}}, W^{\text{out}}, X^{\text{init}} \rangle \in \mathbf{W}^{\text{layer}}_{\mathbf{s}}$ is the MinMax Layer where the output function is $\text{MLP}(W^{\text{out}})$, the reset function of unit i is $\text{MLP}(W^{\text{R}}_i)$, and the set function of unit i is $\text{MLP}(W^{\text{R}}_i)$.

The set of possible sizes of a MinMax RNC is $\mathbf{S}^{\text{rnc}} := (\mathbb{N}^{>0})^8$—the set of 8-tuples of non-zero natural numbers. A size $\mathbf{s} = \langle d_{\text{in}}, d_{\text{out}}, d_{\text{model}}, d_{\text{state}}, n_{\text{units}}, n_{\text{layers}}, d_{\text{mlp}}, n_{\text{mlp}} \rangle \in \mathbf{S}^{\text{rnc}}$ is to be interpreted as follows:

- d_{in} is the input dimension, hence the input dimension of the first layer;
- d_{out} is the output dimension, hence the output dimension of the last layer;
- d_{model} is the dimension of the output of each layer except the last one, and also the input of all layers except the first one;
- d_{state} is the state dimension of each MinMax Recurrent Unit, hence the state degree of the RNC;
- n_{units} is the number of MinMax Recurrent Units in each layer;
- n_{layers} is the number of MinMax Layers;
- d_{mlp} is the dimension of the hidden layers of each MLP;
- n_{mlp} is the number of hidden layers of each MLP.

The weights of a MinMax RNC of size $\mathbf{s}$ are

$$\mathbf{W}^{\text{rnc}}_{\mathbf{s}} := \{ \langle W_0, \ldots, W_{n_{\text{layers}}-1} \rangle \mid W_i \in \mathbf{W}^{\text{layer}}_{\mathbf{s}_i} \},$$

where

$$\begin{aligned} \mathbf{s}_0 &= \langle d_{\text{in}}, d_{\text{model}}, d_{\text{state}}, n_{\text{units}}, d_{\text{mlp}}, n_{\text{mlp}} \rangle, \\ \mathbf{s}_i &= \langle d_{\text{model}}, d_{\text{model}}, d_{\text{state}}, n_{\text{units}}, d_{\text{mlp}}, n_{\text{mlp}} \rangle, \quad \forall i \in [1..n_{\text{layers}} - 2], \\ \mathbf{s}_{n_{\text{layers}}-1} &= \langle d_{\text{model}}, d_{\text{out}}, d_{\text{state}}, n_{\text{units}}, d_{\text{mlp}}, n_{\text{mlp}} \rangle. \end{aligned}$$

The MinMax RNC having weights $W = \langle W_0, \ldots, W_{n_{\text{layers}}-1} \rangle \in \mathbf{W}^{\text{rnc}}_{\mathbf{s}}$, denoted by $\text{RNC}(W)$, is the MinMax RNC $S_0 \ltimes \cdots \ltimes S_{n_{\text{layers}}-1}$ where $S_i = \text{Layer}(W_i)$. *End of Definition 219.*

Definition 220. The computation problem of *evaluating MinMax RNCs* has valid inputs and possible outputs respectively given by

$$I := \bigcup_{\mathbf{s} \in \mathbf{S}^{\text{rnc}}} \bigcup_{T=1}^{\infty} I_{\mathbf{s},T} \qquad \text{with } I_{\mathbf{s},T} := \{ \langle W, U \rangle \mid W \in \mathbf{W}^{\text{rnc}}_{\mathbf{s}},\ U \in \mathbb{R}^{T \times d_{\text{in}}} \},$$

$$O := \bigcup_{d_{\text{out}}=1}^{\infty} \bigcup_{T=1}^{\infty} \qquad \text{with } O_{d_{\text{out}},T} := \{ Y \mid Y \in \mathbb{R}^{T \times d_{\text{out}}} \},$$

and the expected output for input $\langle W, U \rangle \in I_{\mathbf{s},T}$ is the sequence Y satisfying the following, for $S = \text{RNC}(W)$,

$$Y[t] = S(U[0] \cdots U[t]) \quad \forall t \in [0..T-1].$$

For inputs in $I_{\mathbf{s},T}$, we call $\mathbf{s}$ the *RNC size* and T the *input-sequence length* or *number of steps*. *End of Definition 220.*

K.3 Proofs

In this section we prove the theorems of Section 5.

- Theorem 5 is given by Lemma 25 for the sequential case, and by Lemma 28 for the parallel case.
- Theorem 6 is given by its more formal variant Theorem 32.
- Theorem 6 is given by its more formal and more general variant Theorem 42, which also show the complexity without collapsing any quantity describing the size of the RNC, also without fixing the number of hidden layers of MLPs.

K.3.1 Sequential

Lemma 25. *SeqRecEval solves MinMax Recurrence Evaluation in time $O(TN^2)$ for T the number of steps and N the state dimension.*

Proof. Let $\langle X_{\text{init}}, A, B\rangle$ be an input to the problem.

Correctness. We show that SeqRecEval solves the problem.

To distinguish variables of the algorithm from mathematical variables, let us add a hat to the variables of the algorithm. In particular, the input variables are $\langle \hat{X}_{\text{init}}, \hat{A}, \hat{B}\rangle$, and the other variables are $\hat{T}, \hat{N}, \hat{X}, \hat{Z}$ (along with the loop indices). We note that the input variables $\langle \hat{X}_{\text{init}}, \hat{A}, \hat{B}\rangle$ are never modified by the algorithm, and hence we can write $\hat{X}_{\text{init}}$, $\hat{A}$, $\hat{B}$ to denote their values, with no ambiguity. Furthermore, the scalar variables $\hat{T}$, $\hat{N}$ as well as the entries of tensors $\hat{X}$ and $\hat{Z}$ are written at most once and never read before they are written, and hence we can write $\hat{T}, \hat{N}, \hat{X}[t,i], \hat{Z}[t,i,j]$ to denote the values held by the variables, with no ambiguity when such values have been assigned.

Let us define X as the tensor of shape $(T+1, N, N)$ satisfying $X[0] = X_{\text{init}}$ and $X[t] = \circledast(X[t-1], A[t-1], B[t-1])$ for every $t \in [1..T]$.

It suffices to show that $\hat{X}[0:T-1] = X[1:T]$ when Line 11 is executed.

We have that the value of the input variables $\langle \hat{X}_{\text{init}}, \hat{A}, \hat{B}\rangle$ is $\langle X_{\text{init}}, A, B\rangle$. Then, after executing Line 1, the value of $\hat{T}$ is T and the value of $\hat{N}$ is N.

We first show by induction on t from 0 to T that after having executed Line 1–4 and after t iterations of the loop of Line 5 the equalities $\hat{Z}[0:t, N] = X[0:t]$ and $\hat{X}[0:t-1] = X[1:t]$ hold. The base case $t = 0$ holds since, after having executed Line 1–4 and before executing any iteration of the loop, the equality $\hat{Z}[0, N] = X[0] = X_{\text{init}}$ holds by Line 4, and the equality $\hat{X}[0:-1] = X[1:0]$ holds vacously. In the inductive case we have $t > 0$, hence the loop has been executed t times, and we assume by induction that the equalities $\hat{Z}[0:t-1, N] = X[0:t-1]$ and $\hat{X}[0:t-2] = X[1:t-1]$ hold. Letting $i \in [1..N]$, we show that that the i-th iteration of the loop of Line 6 ensures that the equality $\hat{Z}[t, N, 0:i-1] = X[t, 0:i-1]$ holds. We have that right after executing Line 7, $\hat{Z}[t, 0, i-1] = B[t-1, i-1]$. We proceed by a second induction on j from 0 to N, showing that that after j iterations of the loop of Line 8, the following equality holds:

$$\hat{Z}[t, j, i-1] = B[t-1, i-1] \oplus \bigoplus_{\ell=0}^{j-1} \big(A[t-1, i-1, \ell] \odot X[t-1, \ell]\big).$$

In the base case we have $j = 0$, and the equality holds by the assignment of Line 7. The inductive case $j > 0$ holds since

$$\begin{aligned}
&\hat{Z}[t, j, i-1] \\
&= \hat{Z}[t, j-1, i-1] \oplus \big(A[t-1, i-1, j-1] \odot \hat{Z}[t-1, N, j-1]\big) \\
&= \hat{Z}[t, j-1, i-1] \oplus \big(A[t-1, i-1, j-1] \odot X[t-1, j-1]\big) && \text{| first inductive hypothesis} \\
&= B[t-1, i-1] \oplus \left(\bigoplus_{\ell=0}^{j-2} \big(A[t-1, i-1, \ell] \odot X[t-1, \ell]\big)\right) \oplus \big(A[t-1, i-1, j-1] \odot X[t-1, j-1]\big) && \text{| second inductive hypothesis} \\
&= B[t-1, i-1] \oplus \bigoplus_{\ell=0}^{j-1} \big(A[t-1, i-1, \ell] \odot X[t-1, \ell]\big) && \text{| associativity of } \oplus
\end{aligned}$$

Therefore,

$$\begin{aligned}\hat{Z}[t,N] &= \begin{pmatrix} B[t-1,0] \\ \vdots \\ B[t-1,N-1] \end{pmatrix} \oplus \begin{pmatrix} \bigoplus_{j=0}^{N-1} \big(A[t-1,0,j] \odot X[t-1,j]\big) \\ \vdots \\ \bigoplus_{j=0}^{N-1} \big(A[t-1,N-1,j] \odot X[t-1,j]\big) \end{pmatrix} \\ &= B[t-1] \oplus (A[t-1] \otimes X[t-1]) \\ &= (A[t-1] \otimes X[t-1]) \oplus B[t-1] \\ &= X[t].\end{aligned}$$

Hence, by Line 10, we have $\hat{X}[t-1] = \hat{Z}[t,N] = X[t]$, which proves the induction on t. Therefore, $\hat{X}[t-1] = X[t]$ for every $t \in [1..T]$, as required.

Execution time. Since reading and writing a real number, as well as executing an elementary operation on real numbers, take constant time in the assumed model of computation, we have that the execution time is $O(\ell)$ for ℓ the total number of times Line 9 is executed, and hence it is $O(TN^2)$ as claimed. □

Lemma 26. *On every input $\langle W, U\rangle$ for $W \in \mathbf{W}_{\mathbf{s}}^{\text{layer}}$ of size $\mathbf{s} = \langle d_{\text{in}}, d_{\text{mlp}}, n_{\text{mlp}}, d_{\text{out}}, n_{\text{units}}, d_{\text{state}}\rangle$, and $U \in \mathbb{R}^{T \times d_{\text{in}}}$, the output of SeqLayerEval is $S(U)$ for $S = \text{Layer}(W)$, and its execution time is*

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^2 + d_{\text{mlp}} \cdot (d_{\text{in}} + d_{\text{out}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + n_{\text{units}} d_{\text{state}})\big)\Big)$$

Proof. We first show correctness and then complexity. We have that the dynamics D of S are given by a parallel composition $D_1 \parallel \cdots \parallel D_n$ with each D_i a MinMax Recurrent Unit.

Correctness. After Line 1, the values of $T, d_{\text{in}}, n_{\text{units}}, d_{\text{state}}, d_{\text{out}}$ are correctly set according to the shape of U and the definition of $W \in \mathbf{W}_{\mathbf{s}}^{\text{layer}}$ (Definition 219). After Line 2, the values of $W^{\text{R}}, W^{\text{s}}, W^{\text{out}}, X^{\text{init}}$ are correctly set according to the definition of $W \in \mathbf{W}_{\mathbf{s}}^{\text{layer}}$ (Definition 219). Then the algorithm proceeds to computing the states of the MinMax Layer S on the given input U. Each MinMax Unit of the layer has state dimension d_{state}, and hence the tensor allocated in Line 3 suffices to store the initial state and the subsequent states of all units. Let us now consider an iteration of the loop of Lines 4–13 for index $i \in [0..n_{\text{units}} - 1]$. Each iteration is independent from the others, as it does not read values written in other iterations, nor assigns a value to a memory location that is also written in some other iteration. After executing Lines 5–9, we have that, for every $t \in [0..T-1]$, the value of $A_i[t]$ is $R_i(U[t])$ for $R_i = \text{MLP}(W^{\text{R}}[i])$ the reset function of component i of the layer, and the value of $B_i[t]$ is $s_i(U[t])$ for $s_i = \text{MLP}(W^{\text{s}}[i])$ the set function of component i of the layer. After executing Lines 10–12, we have that $X[0, i \cdot d_{\text{state}} : (i+1) \cdot d_{\text{state}}] = X^{\text{init}}[i]$, and after executing Line 13, by Theorem 25 we have that $X[t+1, i \cdot d_{\text{state}} : (i+1) \cdot d_{\text{state}}] = D_i(X^{\text{init}}[i], U[0:t])$ for every $t \in [0..T-1]$. Thus, after the execution of the loop of Lines 4–13 completes, we have that $X[t+1] = D(X^{\text{init}}, U[0:t])$ for every $t \in [0..T-1]$.

Next, let us consider an iteration of the loop of Lines 15–16 for $t \in [0..T-1]$, noting that each iteration is independent from the others. We have that

$$X[t] \parallel X[t+1] \parallel U[t] = \langle D(X^{\text{init}}, U[0:t-1]),\ D(X^{\text{init}}, U[0:t]),\ U[t]\rangle,$$

and hence after executing Line 16, we have

$$Y[t] = h\big(D(X^{\text{init}}, U[0:t-1]),\ D(X^{\text{init}}, U[0:t]),\ U[t]\big)$$

for $h = \text{MLP}(W^{\text{out}})$ the output function of S. Thus, after the execution of the loop of Lines 15–16 completes, we have that $Y = S(U)$ as required.

Complexity. The execution time of the algorithm is asymptotically bounded by the sum of the following execution times:

- Line 8: $O(n_{\text{units}} \cdot T)$ executions of `EvaluateMLP`, where the MLP has input dimension d_{in}, output dimension d_{state}^2, and n_{mlp} hidden layers of dimension $d_{\text{mlp,u}} := \lceil d_{\text{mlp}} / \sqrt{n_{\text{units}}} \rceil$; the time of each evaluation is $O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,i}} d_{\text{state}}^2)$;

- Line 9: $O(n_{\text{units}} \cdot T)$ executions of `EvaluateMLP` where the MLP has input dimension d_{in}, output dimension d_{state}, and n_{mlp} hidden layers of dimension $d_{\text{mlp,u}} := \lceil d_{\text{mlp}}/n_{\text{units}} \rceil$; the time of each evaluation is $O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}})$;
- Line 13: $O(n_{\text{units}})$ executions of SeqRecEval on inputs of dimension d_{state} and length T; its execution time is $O(T d_{\text{state}}^2)$ by Theorem 25;
- Line 16: $O(T)$ executions of `EvaluateMLP`, where the MLP has input dimension $2n_{\text{units}} d_{\text{state}} + d_{\text{in}}$, output dimension d_{out}, and n_{mlp} hidden layers of dimension d_{mlp}; the time of each evaluation is $O\big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)$.

Thus, the execution time is

$$
\begin{aligned}
&O(n_{\text{units}} \cdot T) \cdot O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}^2) + \\
&O(n_{\text{units}} \cdot T) \cdot O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}) + \\
&O(n_{\text{units}}) \cdot O(T d_{\text{state}}^2) + \\
&O(T) \cdot O\big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp}} d_{\text{out}}\big) \\
&= \\
&O\Big(n_{\text{units}} \cdot T \cdot (d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}^2)\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot (d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}})\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot d_{\text{state}}^2\Big) + \\
&O\Big(T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(n_{\text{units}} \cdot T \cdot (d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}^2)\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot d_{\text{state}}^2\Big) + \\
&O\Big(T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot (d_{\text{in}} d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{state}}^2)\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot d_{\text{state}}^2\Big) + \\
&O\Big(T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot (d_{\text{in}} d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{state}}^2) + \\
&\qquad n_{\text{units}} \cdot T \cdot d_{\text{state}}^2 + \\
&\qquad T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot \big(d_{\text{in}} d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{state}}^2 + n_{\text{units}} \cdot d_{\text{state}}^2 + n_{\text{units}} \cdot d_{\text{mlp}} d_{\text{state}} + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^2 + d_{\text{mlp}} \cdot (d_{\text{in}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{out}} + n_{\text{units}} d_{\text{state}})\big)\Big)
\end{aligned}
$$

□

Lemma 27. *On input $\langle W, U\rangle$ for $W \in \mathbf{W}^{\mathrm{rnc}}_{\mathbf{s}}$ of size*

$$\mathbf{s} = \langle d_{\mathrm{in}}, d_{\mathrm{out}}, d_{\mathrm{model}}, d_{\mathrm{state}}, n_{\mathrm{units}}, n_{\mathrm{layers}}, d_{\mathrm{mlp}}, n_{\mathrm{mlp}}\rangle,$$

and $U \in \mathbb{R}^{T\times d_{\mathrm{in}}}$, the output of SeqCascEval is $S(U)$ for $S = \mathrm{RNC}(W)$, and its execution time is

$$O\Big(T \cdot n_{\mathrm{layers}} \cdot \big(n_{\mathrm{units}} \cdot d^2_{\mathrm{state}} + d_{\mathrm{mlp}} \cdot (d_{\mathrm{in}} + d_{\mathrm{model}} + d_{\mathrm{out}} + n_{\mathrm{mlp}} d_{\mathrm{mlp}} + d^2_{\mathrm{state}} + n_{\mathrm{units}} d_{\mathrm{state}})\big)\Big),$$

and for fixed n_{mlp} and $d := \max\{d_{\mathrm{in}}, d_{\mathrm{out}}, d_{\mathrm{mlp}}, n_{\mathrm{units}} \cdot d_{\mathrm{state}}\}$,

$$O\Big(T \cdot n_{\mathrm{layers}} \cdot d \cdot \big(d + d^2_{\mathrm{state}})\big)\Big).$$

Proof. We first show correctness and then complexity. We have $W = \langle W_1, \ldots, W_{n_{\mathrm{layers}}-1}\rangle$ and that the RNC S is given by the cascade composition $S_0 \ltimes \cdots \ltimes S_{n_{\mathrm{layers}}-1}$ where each layer is $S_i = \mathrm{Layer}(W_i)$.

Correctness. After Line 1, the value of n_{layers} is correctly set according to the definition of $W \in \mathbf{W}^{\mathrm{rnc}}_{\mathbf{s}}$ (Definition 219). Right after Line 2, the value of Y is U. Thus, after the i-th iteration of the loop, the value of Y is $S_{0:i}(U)$ for $S_{0:i} = S_0 \ltimes \cdots \ltimes S_i$, by Theorem 26. Therefore, when the execution of the loops completes, the value of Y is $S(U)$ as required.

Complexity. The execution time is given by $O(n_{\mathrm{layers}})$ calls of `SeqLayerEval` to evaluate a layer having input dimension d_{in} and output dimension d_{model} for $i = 0$, input and output dimension d_{model} for each $i \in [1..n_{\mathrm{layers}} - 2]$, and input dimension d_{model} and output dimension d_{out} for $i = n_{\mathrm{layers}} - 1$. For the rest, the size of each layer is described by $\langle d_{\mathrm{state}}, n_{\mathrm{units}}, d_{\mathrm{mlp}}, n_{\mathrm{mlp}}\rangle$.

By Theorem 26, the execution time for the first layer is

$$O\Big(T \cdot \big(n_{\mathrm{units}} \cdot d^2_{\mathrm{state}} + d_{\mathrm{mlp}} \cdot (d_{\mathrm{in}} + d_{\mathrm{model}} + n_{\mathrm{mlp}} d_{\mathrm{mlp}} + d^2_{\mathrm{state}} + n_{\mathrm{units}} d_{\mathrm{state}})\big)\Big),$$

the execution time for each layer $i \in [1..n_{\mathrm{layers}} - 2]$ is

$$O\Big(T \cdot \big(n_{\mathrm{units}} \cdot d^2_{\mathrm{state}} + d_{\mathrm{mlp}} \cdot (d_{\mathrm{model}} + d_{\mathrm{model}} + n_{\mathrm{mlp}} d_{\mathrm{mlp}} + d^2_{\mathrm{state}} + n_{\mathrm{units}} d_{\mathrm{state}})\big)\Big),$$

and the execution time for layer $i = n_{\mathrm{layers}} - 1$ is

$$O\Big(T \cdot \big(n_{\mathrm{units}} \cdot d^2_{\mathrm{state}} + d_{\mathrm{mlp}} \cdot (d_{\mathrm{model}} + d_{\mathrm{out}} + n_{\mathrm{mlp}} d_{\mathrm{mlp}} + d^2_{\mathrm{state}} + n_{\mathrm{units}} d_{\mathrm{state}})\big)\Big).$$

Thus, overall we have

$$O\Big(T \cdot n_{\mathrm{layers}} \cdot \big(n_{\mathrm{units}} \cdot d^2_{\mathrm{state}} + d_{\mathrm{mlp}} \cdot (d_{\mathrm{in}} + d_{\mathrm{model}} + d_{\mathrm{out}} + n_{\mathrm{mlp}} d_{\mathrm{mlp}} + d^2_{\mathrm{state}} + n_{\mathrm{units}} d_{\mathrm{state}})\big)\Big).$$

For fixed n_{mlp} and $d := \max\{d_{\mathrm{in}}, d_{\mathrm{out}}, d_{\mathrm{mlp}}, n_{\mathrm{units}} \cdot d_{\mathrm{state}}\}$,

$$\begin{aligned}
&O\Big(T \cdot n_{\mathrm{layers}} \cdot \big(n_{\mathrm{units}} \cdot d^2_{\mathrm{state}} + d \cdot (d + d^2_{\mathrm{state}})\big)\Big)\\
&= O\Big(T \cdot n_{\mathrm{layers}} \cdot \big(d \cdot d_{\mathrm{state}} + d \cdot (d + d^2_{\mathrm{state}})\big)\Big)\\
&= O\Big(T \cdot n_{\mathrm{layers}} \cdot d \cdot \big(d + d^2_{\mathrm{state}}\big)\Big).
\end{aligned}$$

□

Theorem 32. *MinMax Recurrence Evaluation can be solved sequentially in time $O(TN^2)$ for T the number of steps and N the state dimension.*

Proof. Immediate by Lemma 25. □

K.3.2 Auxiliary

Theorem 33. *On any given input $A \in \mathbb{R}^n$, the output of* `ParallelMin` *(Algorithm 1) is $\bigodot_{i=0}^{n-1} A[i]$, its depth is $O(\log n)$, and its work is $O(n)$.*

Proof. We first show correctness and then complexity.

Correctness. We proceed by induction on n. In the base case $n = 1$ and we have that the function returns $A[0]$ as required (Lines 2–3). In the inductive case we have $n > 1$ and we assume that the output of function `ParallelMin` on input B is $\bigodot_{i=0}^{m-1} B[i]$ for every input B of length $m \leq n-1$. We consider two cases separately, according to whether n is even or odd.

In the first case, n is even. Right after Line 5, we have

$$m = \lfloor n/2 \rfloor = n/2.$$

We have that right after Lines 7–8, for every $i \in [0..m-2]$, the equality $\hat{B}[i] = A[2i] \odot A[2i+1]$ holds. Then, in Lines 9–12, we have that Line 10 is executed, and hence $\hat{B}[m-1] = A[n-2] \odot A[n-1]$. Let us define the following sets of indices

$$\begin{aligned} I_0 &:= \{2i \mid i \in [0..m-2]\} = \{0, 2, \ldots, 2m-4\} = \{0, 2, \ldots, n-4\}, \\ I_1 &:= \{2i+1 \mid i \in [0..m-2]\} = \{1, 3, \ldots, 2m-3\} = \{1, 3, \ldots, n-3\}. \end{aligned}$$

We note that $I_0 \cap I_1 = \emptyset$ and $I_0 \cup I_1 = [0..n-3]$. Then, by Line 13, the value returned by `ParallelMin`(A) is the value returned by `ParallelMin`$(\hat{B})$, which is as follows by the inductive hypothesis since the length of $\hat{B}$ is smaller than n,

$$\begin{aligned} \bigodot_{i=0}^{m-1} \hat{B}[i] &= \hat{B}[m-1] \odot \bigodot_{i=0}^{m-2} \hat{B}[i] \\ &= A[n-2] \odot A[n-1] \odot \bigodot_{i=0}^{m-2} A[2i] \odot A[2i+1] \\ &= A[n-2] \odot A[n-1] \odot \left(\bigodot_{i=0}^{m-2} A[2i] \right) \odot \left(\bigodot_{i=0}^{m-2} A[2i+1] \right) \\ &= A[n-2] \odot A[n-1] \odot \left(\bigodot_{i \in I_0} A[i] \right) \odot \left(\bigodot_{i \in I_1} A[i] \right) \\ &= \bigodot_{i=0}^{n-1} A[i], \end{aligned}$$

as required.

In the second case n is odd, hence $n = n' + 1$ with n' even. Right after Line 5, we have

$$m = \lfloor n/2 \rfloor = \lfloor n'/2 + 1/2 \rfloor = n'/2.$$

We have that right after Lines 7–8, for every $i \in [0..m-2]$, the equality $\hat{B}[i] = A[2i] \odot A[2i+1]$ holds. Then, in Lines 9–12, we have that Line 12 is executed, and hence $\hat{B}[m-1] = A[n-3] \odot A[n-2] \odot A[n-1]$. Let us define the following sets of indices

$$\begin{aligned} I_0 &:= \{2i \mid i \in [0..m-2]\} = \{0, 2, \ldots, 2m-4\} = \{0, 2, \ldots, n'-4\} = \{0, 2, \ldots, n-5\}, \\ I_1 &:= \{2i+1 \mid i \in [0..m-2]\} = \{1, 3, \ldots, 2m-3\} = \{1, 3, \ldots, n'-3\} = \{1, 3, \ldots, n-4\}. \end{aligned}$$

We note that $I_0 \cap I_1 = \emptyset$ and $I_0 \cup I_1 = [0..n-4]$. Then, by Line 13, the value returned by `ParallelMin`(A) is the value returned by `ParallelMin`$(\hat{B})$, which is as follows by the inductive

hypothesis since the length of B is smaller than n,

$$\begin{aligned}
\bigodot_{i=0}^{m-1} \hat{B}[i] &= \hat{B}[m-1] \odot \bigodot_{i=0}^{m-2} \hat{B}[i] \\
&= A[n-3] \odot A[n-2] \odot A[n-1] \odot \bigodot_{i=0}^{m-2} A[2i] \odot A[2i+1] \\
&= A[n-3] \odot A[n-2] \odot A[n-1] \odot \left(\bigodot_{i=0}^{m-2} A[2i]\right) \odot \left(\bigodot_{i=0}^{m-2} A[2i+1]\right) \\
&= A[n-3] \odot A[n-2] \odot A[n-1] \odot \left(\bigodot_{i \in I_0} A[i]\right) \odot \left(\bigodot_{i \in I_1} A[i]\right) \\
&= \bigodot_{i=0}^{n-1} A[i],
\end{aligned}$$

as required.

Complexity. The depth is given by

$$\operatorname{depth}(n) = \begin{cases} O(1) & \text{if } n = 1, \\ \operatorname{depth}(\lfloor n/2 \rfloor) + O(1) & \text{otherwise.} \end{cases}$$

The work is given by

$$\operatorname{work}(n) = \begin{cases} O(1) & \text{if } n = 1, \\ \operatorname{work}(\lfloor n/2 \rfloor) + O(n) & \text{otherwise.} \end{cases}$$

Therefore, $\operatorname{depth}(n) = O(\log n)$ and $\operatorname{work}(n) = O(n)$ by the Master Theorem, cf. [62]. □

Theorem 34. *On any given input $A \in \mathbb{R}^n$, the output of* `ParallelMax` *(Algorithm 2) is $\bigoplus_{i=0}^{n-1} A[i]$, its depth is $O(\log n)$, and its work is $O(n)$.*

Proof. Omitted as it is analogous to `ParallelMin` (Theorem 33). □

Theorem 35. *On any given input $A, B \in \mathbb{R}^{m \times n}$, the output of* `ParallelAdd` *(Algorithm 3) is $C = A \oplus B$, its depth is $O(1)$, and its work is $O(m \cdot n)$.*

Proof. Correctness is immediate as the algorithm closely follows the definition of $A \oplus B$. Depth is $O(1)$ since the algorithm allocates C (Line 2) which has constant cost, and then it executes Line 5 in parallel of each value of i, j. Work is $O(m \cdot n)$ since the algorithm allocates C (Line 2) which has constant cost, and then it executes Line 5 for each value of $i \in [0..m-1]$ and $j \in [0..n-1]$, and hence it executes Line 5 $m \cdot n$ times overall. □

Theorem 36. *On any given input $A \in \mathbb{R}^{m \times p}, B \in \mathbb{R}^{p \times n}$, the output of* `ParallelMult` *(Algorithm 4) is $C = A \otimes B$, its depth is $O(\log p)$, and its work is $O(m \cdot n \cdot p)$.*

Proof. We first show correctness and the complexity.

Correctness. We have that, for each $i \in [0..m-1]$, each $j \in [0..n-1]$, and each $k \in [0..p-1]$, after executing Line 7, it holds that

$$T[i, j, k] = A[i, k] \odot B[k, j].$$

Thus, by Theorem 34, for each $i \in [0..m-1]$ and each $j \in [0..n-1]$, after executing Line 8, it holds that

$$C[i, j] = \bigoplus_{k=0}^{p-1} T[i, j, k] = \bigoplus_{k=0}^{p-1} (A[i, k] \odot B[k, j]),$$

which is the required value for $C[i, j]$.

Complexity. The depth is $O(1)$ plus the maximum depth of `ParallelMax` on an input $T[i,j]$. Since each $T[i,j]$ has length p, by Theorem 34 we have that the depth of each call of `ParallelMax` has depth $O(\log p)$. Thus, the algorithm has depth $O(\log p)$ overall.

The work is $O((m \cdot n) \cdot (c + w))$ for c the cost of Line 7 (which is executed $m \cdot n$ times) and for w the maximum work of a call of `ParallelMax` on input $T[i,j]$. Since each $T[i,j]$ has length p, by Theorem 34 we have that the work of each call of `ParallelMax` is $O(p)$. Thus, since c is constant, the algorithm has work $O(m \cdot n \cdot p)$ overall. □

Theorem 37. *On any given input $(A_1, B_1) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$, $(A_2, B_2) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$, the output of* `ParallelComp` *(Algorithm 5) is $\langle A, B\rangle = \langle A_1, B_1\rangle \circledast \langle A_2, B_2\rangle$, its depth is $O(\log n)$, and its work is $O(n^3)$.*

Proof. We first show correctness and the complexity.

Correctness. By Theorem 36, after executing Line 4 it holds that $A = A_2 \otimes A_1$, and after executing Line 5 it holds that $Z = A_2 \otimes B_1$. Then, by Theorem 35, after executing Line 6 it holds that $B = Z \oplus B_2 = (A_2 \otimes B_1) \oplus B_2$. Thus, the returned pair (A, B) is as required.

Complexity. The depth is $O(d_1 + d_2 + d_3)$ for d_1 the depth of `ParallelMult`(A_2, A_1), d_2 the depth of `ParallelMult`(A_2, B_1), and d_3 the depth of `ParallelAdd`(Z, B_2). By Theorem 36, we have that d_1 and d_2 are $O(\log n)$. By Theorem 35, we have that d_3 is $O(1)$. Therefore, the algorithm has depth $O(\log n + \log n + 1) = O(\log n)$.

The work is $O(w_1 + w_2 + w_3)$ for w_1 the work of `ParallelMult`(A_2, A_1), w_2 the work of `ParallelMult`(A_2, B_1), and w_3 the work of `ParallelAdd`(Z, B_2). By Theorem 36, we have that w_1 is $O(n^3)$ and w_2 is $O(n^2)$. By Theorem 35, we have that w_3 is $O(n)$. Therefore, the algorithm has work $O(n^3 + n^2 + n) = O(n^3)$. □

Theorem 38. *On any given input $(A, B, X) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n \times \mathbb{R}^n$, the output of* `ParallelApply` *(Algorithm 6) is $C = (A \otimes X) \oplus B$, its depth is $O(\log n)$, and its work is $O(n^2)$.*

Proof. We first show correctness. After Line 2 we have that $Z = A \otimes X$ by Theorem 36. Hence by Theorem 35, after Line 3 $Y = Z \oplus B = (A \otimes X)\oplus$ as required.

Next we analyse its complexity. The depth is $O(\log n)$ since it is the sum of the following depths:

- the depth of `ParallelMult` on input (A, X), which is $O(\log n)$ by Theorem 36;
- the depth of `ParallelAdd` on input (Z, B), which is $O(1)$ by Theorem 35.

The work is $O(n^2)$ since it is the sum of the following work:

- the work of `ParallelMult` on input (A, X), which is $O(n^2)$ by Theorem 36;
- the work of `ParallelAdd` on input (Z, B), which is $O(n)$ by Theorem 35.

□

K.3.3 Parallel

Definition 211. A *MinMax-recurrence pair* over $V \subseteq \mathbb{R}$ of dimension $n \in \mathbb{N}^{>0}$ is a pair $M = \langle A, b\rangle$ with $A \in V^{n\times n}$, $b \in V^n$, and its *MinMax-recurrence map* is $\mathcal{A}_M(x) := (A \otimes x) \oplus b$. MinMax-recurrence pairs $M_1 = \langle A_1, b_1\rangle$ and $M_2 = \langle A_2, b_2\rangle$ are *composable* if they have the same dimension; in this case, their *MinMax-recurrence composition* is $M_1 \circledast M_2 := \langle A_2 \otimes A_1, (A_2 \otimes b_1) \oplus b_2\rangle$.

Definition 221. For $n \in \mathbb{N}^{>0}$ and for V a compact subset $V \subset \mathbb{R}$, we define the *MinMax monoid over V of dimension n* as $\mathcal{M}_{V,n} := \langle V^{n\times n} \times V^n, \circledast\rangle$. We drop the subscripts V and n, writing $\mathcal{M}$ instead of $\mathcal{M}_{V,n}$, when specifying them is not relevant or they are clear from the context.

Proposition 35. *Every MinMax monoid $\mathcal{M}_{V,n}$ is a monoid with identity element $\mathcal{E} = \langle E, e\rangle$ given as:*

$$e[i] = \min(V), \qquad E[i,j] = \begin{cases} \max(V) & \text{if } i = j, \\ \min(V) & \text{if } i \neq j. \end{cases}$$

Namely, for all $M_1, M_2, M_2 \in \mathcal{M}_{V,n}$, the following properties hold:

(1) Closedness: $M_1 \circledast M_2 \in \mathcal{M}_{V,n}$;

(2) Associativity: $\big(M_1 \circledast M_2\big) \circledast M_3 \;=\; M_1 \circledast \big(M_2 \circledast M_3\big)$;

(3) Identity: $\mathcal{E} \circledast M_1 = M_1 \circledast \mathcal{E} = M_1$.

Proof. We show the three properties separately.

Closedness. We show Property (1). Let $M_1 = \langle A_1, b_1\rangle$ and let $M_2 = \langle A_2, b_2\rangle$. Let us note that $A_1, A_2 \in V^{n\times n}$ and $b_1, b_2 \in V^n$ with V a compact subset $V \subset \mathbb{R}$. We have

$$M_1 \circledast M_2 = \langle A_1, b_1\rangle \circledast \langle A_2, b_2\rangle = \langle A_2 \otimes A_1, (A_2 \otimes b_1) \oplus b_2\rangle.$$

Then $C = A_2 \otimes A_1 \in V^{n\times n}$ since the operator $\otimes$ yields an n-by-n matrix C when applied to a pair of n-by-n matrices A_2, A_1, and the elements of C are elements occurring in A_2, A_1 (hence in V) since they are obtained via min and max operations.

Associativity. We show Property (2). Let $M_1 = \langle A_1, b_1\rangle$, $M_2 = \langle A_2, b_2\rangle$, $M_3 = \langle A_3, b_3\rangle$. The left-hand side of the identity can be equivalently rewritten as follows:

$$\begin{aligned}
&\big(\langle A_1, b_1\rangle \circledast \langle A_2, b_2\rangle\big) \circledast \langle A_3, b_3\rangle \\
&= \Big\langle A_2 \otimes A_1,\ (A_2 \otimes b_1) \oplus b_2 \Big\rangle \circledast \langle A_3, b_3\rangle && \text{| def. of } \circledast \\
&= \Big\langle A_3 \otimes (A_2 \otimes A_1),\ \Big(A_3 \otimes \big((A_2 \otimes b_1) \oplus b_2\big)\Big) \oplus b_3 \Big\rangle && \text{| def. of } \circledast \\
&= \Big\langle (A_3 \otimes A_2) \otimes A_1,\ \Big(A_3 \otimes \big((A_2 \otimes b_1) \oplus b_2\big)\Big) \oplus b_3 \Big\rangle && \text{| associativity of } \otimes \\
&= \Big\langle (A_3 \otimes A_2) \otimes A_1,\ \Big(\big(A_3 \otimes (A_2 \otimes b_1)\big) \oplus (A_3 \otimes b_2)\Big) \oplus b_3 \Big\rangle && \text{| distributivity of } \otimes \text{ over } \oplus \\
&= \Big\langle (A_3 \otimes A_2) \otimes A_1,\ \big(A_3 \otimes (A_2 \otimes b_1)\big) \oplus \big((A_3 \otimes b_2) \oplus b_3\big) \Big\rangle && \text{| associativity of } \oplus \\
&= \Big\langle (A_3 \otimes A_2) \otimes A_1,\ \big((A_3 \otimes A_2) \otimes b_1\big) \oplus \big((A_3 \otimes b_2) \oplus b_3\big) \Big\rangle, && \text{| associativity of } \otimes
\end{aligned}$$

and the right-hand side of the identity can be equivalently rewritten as follows:

$$\begin{aligned}
&\langle A_1, b_1\rangle \circledast \big(\langle A_2, b_2\rangle \circledast \langle A_3, b_3\rangle\big) \\
&= \langle A_1, b_1\rangle \circledast \Big\langle A_3 \otimes A_2,\ (A_3 \otimes b_2) \oplus b_3 \Big\rangle && \text{| def. of } \circledast \\
&= \Big\langle (A_3 \otimes A_2) \otimes A_1,\ \big((A_3 \otimes A_2) \otimes b_1\big) \oplus \big((A_3 \otimes b_2) \oplus b_3\big) \Big\rangle. && \text{| def. of } \circledast
\end{aligned}$$

The two expressions we have derived coincide, and hence associativity is proved.

Identity element. We show Property (3). Let $M_1 = \langle A, b\rangle$, let $\mathcal{E} = \langle E, e\rangle$, and let $v_{\min} := \min(V), v_{\max} := \max(V)$.

First we show $M_1 \circledast \mathcal{E} = M_1$. We have

$$\langle A, b\rangle \circledast \langle E, e\rangle = \big\langle E \otimes A,\ (E \otimes b) \oplus e\big\rangle.$$

Then, for every $i, j \in [1..n]$,

$$\begin{aligned}(E \otimes A)_{i,j} &= \bigoplus_{k=1}^{n} E_{i,k} \odot A_{k,j} \\ &= (E_{i,i} \odot A_{i,j}) \oplus \bigoplus_{k \neq i} E_{i,k} \odot A_{k,j} \\ &= (v_{\max} \odot A_{i,j}) \oplus \bigoplus_{k \neq i} v_{\min} \odot A_{k,j} \\ &= A_{i,j} \oplus \bigoplus_{k \neq i} v_{\min} \\ &= A_{i,j} \oplus v_{\min} \\ &= A_{i,j}.\end{aligned}$$

Thus, $E \otimes A = A$.

Then, for every $i \in [1..n]$,

$$\begin{aligned}(E \otimes b)_i &= \bigoplus_{k=1}^{n} E_{i,k} \odot b_k \\ &= (E_{i,i} \odot b_i) \oplus \bigoplus_{k \neq i} E_{i,k} \odot b_k \\ &= (v_{\max} \odot b_i) \oplus \bigoplus_{k \neq i} v_{\min} \odot b_k \\ &= b_i \oplus \bigoplus_{k \neq i} v_{\min} \\ &= b_i \oplus v_{\min} \\ &= b_i,\end{aligned}$$

and hence

$$\begin{aligned}((E \otimes b) \oplus e)_i &= b_i \oplus e_i \\ &= b_i \oplus v_{\min} \\ &= b_i.\end{aligned}$$

Thus, $(E \otimes b) \oplus e = b$. Overall we have $\langle A, b\rangle \circledast \langle E, e\rangle = \langle A, b\rangle$ as required.

Next we show $\mathcal{E} \circledast M_1 = M_1$. We have

$$\langle E, e\rangle \circledast \langle A, b\rangle = \big\langle A \otimes E,\ (A \otimes e) \oplus b\big\rangle.$$

Then, for every $i, j \in [1..n]$,

$$\begin{aligned}(A \otimes E)_{i,j} &= \bigoplus_{k=1}^{n} A_{i,k} \odot E_{k,j} \\ &= (A_{i,j} \odot E_{j,j}) \oplus \bigoplus_{k \neq j} A_{i,k} \odot E_{k,j} \\ &= (A_{i,j} \odot v_{\max}) \oplus \bigoplus_{k \neq j} A_{i,k} \odot v_{\min} \\ &= A_{i,j} \oplus \bigoplus_{k \neq j} v_{\min} \\ &= A_{i,j} \oplus v_{\min} \\ &= A_{i,j}.\end{aligned}$$

Thus, $A \otimes E = A$.

Then, for every $i \in [1..n]$,

$$\begin{aligned}(A \otimes e)_i &= \bigoplus_{k=1}^{n} A_{i,k} \odot e_k \\ &= \bigoplus_{k=1}^{n} A_{i,k} \odot v_{\min} \\ &= \bigoplus_{k=1}^{n} v_{\min} \\ &= v_{\min}\end{aligned}$$

and hence

$$\begin{aligned}((A \otimes e) \oplus b)_i &= v_{\min} \oplus b_i \\ &= b_i.\end{aligned}$$

Thus, $(A \otimes e) \oplus b = b$. Overall we have $\langle E, e\rangle \circledast \langle A, b\rangle = \langle A, b\rangle$ as required, and, together with the previous point, it proves the property. □

Definition 222. Let $n \in \mathbb{N}^{>0}$, let V be a compact subset $V \subset \mathbb{R}$, let $\mathcal{M}_{V,n} = \langle V^{n\times n} \times V^n, \circledast\rangle$ be the resulting monoid, and let $\langle E, e\rangle$ be the identity of $\mathcal{M}_{V,n}$. The *(default) action* of a MinMax monoid $\mathcal{M}_{V,n}$ is the function $\tau : (V^{n\times n} \times V^n) \times V^n \to V^n$ defined as $\tau(A, b, x) := (A \otimes x) \oplus b$. We also write $\tau_{A,b}(x)$ for $\tau(A, b, x)$.

Proposition 36. *For every MinMax monoid $\mathcal{M}_{V,n}$, the default action τ of $\mathcal{M}_{V,n}$ is a monoid action; namely, it satisfies the following properties:*

(1) for all $M_1, M_2 \in \mathcal{M}_{V,n}$, we have $\tau_{M_1} \circ \tau_{M_2} = \tau_M$ for $M = M_1 \circledast M_2$;

(2) $\tau_{\mathcal{E}}$ is the identity function—i.e., the equality $\tau_{\mathcal{E}}(x) = x$ holds for all $x \in V^n$.

Proof. We show Property (1) and Property (2) separately.

Property (1). We show Property (1). Let $M_1 = \langle A_1, b_1\rangle$, let $M_2 = \langle A_2, b_2\rangle$, let $M = \langle A, b\rangle$, and let $x \in V^n$. It suffices to show $\tau_{M_2}(\tau_{M_1}(x)) = \tau_M(x)$. First we have

$$M = \langle A, b\rangle = \langle A_1, b_1\rangle \circledast \langle A_2, b_2\rangle = \langle A_2 \otimes A_1, (A_2 \otimes b_1) \oplus b_2\rangle,$$

and hence

$$A = A_2 \otimes A_1, \tag{44}$$
$$b = (A_2 \otimes b_1) \oplus b_2. \tag{45}$$

Then we have

$$\begin{aligned}
\tau_{M_2}(\tau_{M_1}(x)) &= \tau_{A_2,b_2}(\tau_{A_1,b_1}(x)) && \text{| def. of } M_1, M_2\\
&= \tau_{A_2,b_2}\big((A_1 \otimes x) \oplus b_1\big) && \text{| def. of } \tau_{A_1,b_1}\\
&= \Big(A_2 \otimes \big((A_1 \otimes x) \oplus b_1\big)\Big) \oplus b_2 && \text{| def. of } \tau_{A_2,b_2}\\
&= \Big(\big(A_2 \otimes (A_1 \otimes x)\big) \oplus (A_2 \otimes b_1)\Big) \oplus b_2 && \text{| distributivity of } \otimes \text{ over } \oplus\\
&= \Big(\big((A_2 \otimes A_1) \otimes x\big) \oplus (A_2 \otimes b_1)\Big) \oplus b_2 && \text{| associativity of } \otimes\\
&= \big((A_2 \otimes A_1) \otimes x\big) \oplus \Big((A_2 \otimes b_1) \oplus b_2\Big) && \text{| associativity of } \oplus\\
&= \big(A \otimes x\big) \oplus \Big((A_2 \otimes b_1) \oplus b_2\Big) && \text{| by Eq. (44)}\\
&= \big(A \otimes x\big) \oplus b && \text{| by Eq. (45)}\\
&= \tau_{A,b}(x) && \text{| def. of } \tau_{A,b}\\
&= \tau_M(x), && \text{| def. of } \tau_M
\end{aligned}$$

as required.

Property (2). We show Property (2).

Let $\mathcal{E} = \langle E, e\rangle$, let $v_{\min} := \min(V)$, let $v_{\max} := \max(V)$, and let $x \in V^n$.

It suffices to show $\tau_{\mathcal{E}}(x) = x$.

By Proposition 35, we have $E[i,i] = v_{\max}$ for all $i \in [0..n)$, $E[i,j] = v_{\max}$ for all $i, j \in [0..n)$ with $i \neq j$, and $e[i] = v_{\min}$ for all $i \in [0..n)$.

Then, we have

$$\begin{aligned}
\tau_{\mathcal{E}}(x) &= \tau_{E,e}(x) && \text{| def. of } \mathcal{E}\\
&= (E \otimes x) \oplus e. && \text{| def. of } \tau_{E,e}
\end{aligned}$$

Then, for each $i \in [0..n)$,

$$\begin{aligned}
\Big((E \otimes x) \oplus e\Big)_i &= \left(\bigoplus_{j=0}^{n-1} E[i,j] \odot x[j]\right) \oplus e[i] \\
&= (E[i,i] \odot x[i]) \oplus \left(\bigoplus_{j \in [0..n): j \neq i} E[i,j] \odot x[j]\right) \oplus e[i] \\
&= (v_{\max} \odot x[i]) \oplus \left(\bigoplus_{j \in [0..n): j \neq i} v_{\min} \odot x[j]\right) \oplus e[i] && \text{| def. of } E \\
&= (v_{\max} \odot x[i]) \oplus \left(\bigoplus_{j \in [0..n): j \neq i} v_{\min} \odot x[j]\right) \oplus v_{\min} && \text{| def. of } e \\
&= x[i] \oplus \left(\bigoplus_{j \in [0..n): j \neq i} v_{\min} \odot x[j]\right) \oplus v_{\min} && \text{| } v_{\max} \geq x[i] \\
&= x[i] \oplus \left(\bigoplus_{j \in [0..n): j \neq i} v_{\min}\right) \oplus v_{\min} && \text{| } v_{\min} \leq x[j] \\
&= x[i] \oplus v_{\min} \oplus v_{\min} \\
&= x[i]. && \text{| } v_{\min} \leq x[i]
\end{aligned}$$

Thus, $\Big((E \otimes x) \oplus e\Big) = x$, and hence by the above we have shown $\tau_{\mathcal{E}}(x) = x$, as required. □

Theorem 39. *On input $(v_{\min}, v_{\max}, N)$, the output of* `ParallelIdentity` *(Algorithm 7) is the identity element $\mathcal{E}$ of the MinMax monoid $\mathcal{M}_{[v_{\min}, v_{\max}], N}$, its depth is $O(1)$ and its work is $O(N^2)$.*

Proof. Correctness is immediate as the algorithm closely follows the characterisation of $\mathcal{E}$ as given in Proposition 35. The depth is $O(1)$ as the algorithm consists of two subsequent parallel blocks, at Lines 3–6 and Lines 7–9, respectively; and each parallel block consists of $O(1)$ instructions executed in parallel. The work is $O(N^2)$ as the algorithm performs work $O(N^2)$ to assign values to the matrix E of shape (N, N) (Lines 3–6) and work $O(N)$ to assign values to the array e of length N (Lines 7–9). □

Theorem 40. *Let $v_{\min}, v_{\max} \in \mathbb{R}$ with $v_{\min} \leq v_{\max}$, let $V = [v_{\min}, v_{\max}]$, let $A \in V^{T \times N \times N}$, and let $B \in V^{T \times N}$. On input $\langle A, B, v_{\min}, v_{\max} \rangle$, the output of* `ParallelScan` *(Algorithm 8) is (C, D) with*

$$(C[t], D[t]) = (A[0], B[0]) \circledast \cdots \circledast (A[t-1], B[t-1]) \circledast \mathcal{E}, \quad \forall t \in [0..T),$$

for $\mathcal{E}$ the identity element of the MinMax monoid $\mathcal{M}_{V,N}$; furthermore, its depth is $O(\log T \cdot \log N)$, and its work is $O(TN^3)$.

Proof. We show correctness and then complexity.

Correctness. We proceed by induction on T. In the base case $T = 1$ and the algorithm executes Line 4 returning the identity element of the MinMax monoid $\mathcal{M}_{[v_{\min}, v_{\max}], N}$ by Theorem 39.

In the inductive case we have $T > 1$ and we assume that `ParallelScan`$(X, Y, v_{\min}, v_{\max})$ returns the expected output on all inputs $\langle X, Y, v_{\min}, v_{\max} \rangle$ of length at most $T - 1$.

We consider two cases separately, depending on whether T is even or odd.

In the first case we assume that T is even, and hence $m = \ell = T/2$ after executing Line 5. Let $M_t = \langle A[t], B[t] \rangle$ for each $t \in [0..T-1]$. We have that, after executing Line 8–9, for every

$t \in [0..m-1]$, by Theorem 37, it holds that

$$\langle X[t], Y[t] \rangle = M_{2t} \circledast M_{2t+1}. \tag{46}$$

Lines 10–11 are not executed since T is even. After executing Line 12, by the inductive hypothesis (IH) we have that, for every $t \in [0..m-1] = [0..(T/2)-1]$,

$$\begin{aligned}
\langle Z[t], W[t] \rangle &= \langle X[0], Y[0] \rangle \circledast \cdots \circledast \langle X[t-1], Y[t-1] \rangle \circledast \mathcal{E} && \text{| Line 12 and IH} && (47)\\
&= \big(M_0 \circledast M_1\big) \circledast \cdots \circledast \big(M_{2(t-1)} \circledast M_{2(t-1)+1}\big) \circledast \mathcal{E} && \text{| Eq. (46)} && (48)\\
&= M_0 \circledast \cdots \circledast M_{2t-1} \circledast \mathcal{E}. && \text{| associativity} && (49)
\end{aligned}$$

Then, after executing Lines 13–19, for each even $t \in [0..T-2]$, it holds that

$$\begin{aligned}
\langle C[t], D[t] \rangle &= \langle Z[t/2], W[t/2] \rangle && \text{| Line 17} && (50)\\
&= M_0 \circledast \cdots \circledast M_{2(t/2)-1} \circledast \mathcal{E} && \text{| Eq. (49) as } t/2 \in [0..(T/2)-1] && (51)\\
&= M_0 \circledast \cdots \circledast M_{t-1} \circledast \mathcal{E}, && \text{| } 2(t/2)-1 = t-1 && (52)
\end{aligned}$$

and for each odd $t \in [1..T-1]$, it holds that

$$\begin{aligned}
\langle C[t], D[t] \rangle &= \langle Z[(t-1)/2], W[(t-1)/2] \rangle \circledast M_{t-1} && \text{| Line 19 and Thm. 37} && (53)\\
&= M_0 \circledast \cdots \circledast M_{2((t-1)/2)-1} \circledast \mathcal{E} \circledast M_{t-1} && \text{| Eq. (49) as } (t-1)/2 \in [0..(T/2)-1] && (54)\\
&= M_0 \circledast \cdots \circledast M_{t-2} \circledast \mathcal{E} \circledast M_{t-1} && \text{| } 2((t-1)/2)-1 = t-2 && (55)\\
&= M_0 \circledast \cdots \circledast M_{t-2} \circledast M_{t-1} && \text{| As } \mathcal{E} \text{ is identity and } t-1 \geq 0 && (56)\\
&= M_0 \circledast \cdots \circledast M_{t-1} \circledast \mathcal{E}. && \text{| As } \mathcal{E} \text{ is identity and } t-1 \geq 0 && (57)
\end{aligned}$$

Therefore, at the return Line 20, by Equations (52) and (57), for each $t \in [0..T-1]$, it holds that

$$\langle C[t], D[t] \rangle = M_0 \circledast \cdots \circledast M_{t-1} \circledast \mathcal{E},$$

as required.

In the second case we have that T is odd. After executing Line 5, it holds that $m = \lfloor T/2 \rfloor$ and $\ell = \lceil T/2 \rceil$. We have that $T' := T-1$ is even, and hence

$$\begin{aligned}
m &= \lfloor T/2 \rfloor = \lfloor (T'+1)/2 \rfloor = \lfloor T'/2 + 1/2 \rfloor = T'/2 = (T-1)/2,\\
\ell &= \lceil T/2 \rceil = \lceil (T'+1)/2 \rceil = \lceil T'/2 + 1/2 \rceil = T'/2 + 1 = (T-1)/2 + 1.
\end{aligned}$$

In particular, we have

$$m = (T-1)/2, \qquad m-1 = (T-3)/2, \qquad m = \ell - 1. \tag{58}$$

Let $M_t = \langle A[t], B[t] \rangle$ for each $t \in [0..T-1]$. We have that, after executing Lines 8–9, for every $t \in [0..m-1] = [0..(T-3)/2]$, by Theorem 37, it holds that

$$\langle X[t], Y[t] \rangle = M_{2t} \circledast M_{2t+1}. \tag{59}$$

Then, since T is odd, we have that Line 11 is executed, yielding

$$\langle X[m], Y[m] \rangle = M_{T-1}. \tag{60}$$

since $m = \ell - 1$ as noted in Equation (58). Then, after executing Line 12, by the inductive hypothesis (IH) we have that, for every $t \in [0..m] = [0..(T-1)/2]$,

$$\begin{aligned}
&\langle Z[t], W[t] \rangle && && (61)\\
&= \langle X[0], Y[0] \rangle \circledast \cdots \circledast \langle X[t-1], Y[t-1] \rangle \circledast \mathcal{E} && \text{| Line 12 and IH} && (62)\\
&= \big(M_0 \circledast M_1\big) \circledast \cdots \circledast \big(M_{2(t-1)} \circledast M_{2(t-1)+1}\big) \circledast \mathcal{E} && \text{| Eq. (59) as } [0..t-1] \subseteq [0..(T-3)/2] && (63)\\
&= M_0 \circledast \cdots \circledast M_{2t-1} \circledast \mathcal{E}. && \text{| associativity} && (64)
\end{aligned}$$

Then, after executing Lines 13–19, for each even $t \in [0..T-1]$, it holds that

$$\begin{aligned}
\langle C[t], D[t] \rangle &= \langle Z[t/2], W[t/2] \rangle && \text{| Line 17} && (65)\\
&= M_0 \circledast \cdots \circledast M_{2(t/2)-1} \circledast \mathcal{E} && \text{| Eq. (64) as } t/2 \in [0..(T-1)/2] && (66)\\
&= M_0 \circledast \cdots \circledast M_{t-1} \circledast \mathcal{E}, && \text{| } 2(t/2)-1 = t-1 && (67)
\end{aligned}$$

and for each odd $t \in [1..T-2]$, it holds that

$$\langle C[t], D[t]\rangle = \langle Z[(t-1)/2], W[(t-1)/2]\rangle \circledast M_{t-1} \quad \text{| Line 19 and Thm. 37} \tag{68}$$

$$= M_0 \circledast \cdots \circledast M_{2((t-1)/2)-1} \circledast \mathcal{E} \circledast M_{t-1} \quad \text{| Eq. (64) as } (t-1)/2 \in [0..(T-3)/2] \tag{69}$$

$$= M_0 \circledast \cdots \circledast M_{t-2} \circledast \mathcal{E} \circledast M_{t-1} \quad \text{| } 2((t-1)/2) - 1 = t-2 \tag{70}$$

$$= M_0 \circledast \cdots \circledast M_{t-1} \quad \text{| As } \mathcal{E} \text{ is identity and } t > 0 \tag{71}$$

$$= M_0 \circledast \cdots \circledast M_{t-1} \circledast \mathcal{E}. \quad \text{| As } \mathcal{E} \text{ is identity and } t > 0 \tag{72}$$

Therefore, at the return instruction of Line 20, by Equations (67) and (72), for each $t \in [0..T-1]$, it holds that

$$\langle C[t], D[t]\rangle = M_0 \circledast \cdots \circledast M_{t-1} \circledast \mathcal{E},$$

as required. This completes the correctness proof.

Complexity. We first analyse the depth. When $T = 1$, the depth is given by the depth of $\texttt{ParallelIdentity}(v_{\min}, v_{\max}, N)$ which is $O(1)$ by Theorem 39. When $T > 1$, the depth is asymptotically bounded by the depth of the recursive call at Line 12 on inputs of length $\lceil T/2\rceil$ and dimension N, plus the depth of $\texttt{ParallelComp}$ on inputs of dimension N, which is $O(\log N)$ by Theorem 37. Thus, there exist constants a_1, c, c_1 such that the depth is bounded as follows:

$$\operatorname{depth}(T, N) \leq \begin{cases} c & \text{if } T = 1, \\ \operatorname{depth}(\lceil T/2\rceil, N) + c_1 + a_1 \log_2(N+1) & \text{otherwise.} \end{cases} \tag{73}$$

We show $\operatorname{depth}(T, N) \in O(\log T \cdot \log N)$ by showing that, for $a := 3(a_1 + c_1)$, it holds that

$$\operatorname{depth}(T, N) \leq c + a \cdot \log_2 T \cdot \log_2(N+1). \tag{74}$$

The base case $T = 1$ is proved as follows:

$$\operatorname{depth}(T, N) \leq c \quad \text{| by Eq. (73)}$$

$$= c + a \cdot \log_2 T \cdot \log_2(N+1). \quad \text{| } \log_2 T = 0 \text{ as } T = 1$$

In the inductive case we have $T > 1$ and we assume that, for all $T' \leq T-1$,

$$\operatorname{depth}(T', N) \leq c + a \cdot \log_2 T' \cdot \log_2(N+1). \tag{75}$$

We consider two cases separately, according to whether T is even or odd. Let us first consider the case when T is even. We have

$$\operatorname{depth}(T, N) \leq \operatorname{depth}(\lceil T/2\rceil, N) + c_1 + a_1 \log_2(N+1) \quad \text{| Eq. (73)}$$

$$\leq \operatorname{depth}(T/2, N) + c_1 + a_1 \log_2(N+1) \quad \text{| } T \text{ is even}$$

$$\leq c + a \cdot \log_2(T/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1). \quad \text{| Eq. (75)}$$

Thus, it suffices to show that the following inequality holds:

$$c + a \cdot \log_2(T/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1) \leq c + a \cdot \log_2 T \cdot \log_2(N+1).$$

We have

$$c + a \cdot \log_2(T/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1) \leq c + a \cdot \log_2 T \cdot \log_2(N+1)$$

$$\text{if} \quad a \cdot \log_2(T/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1)$$

$$\text{if} \quad c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) - a \cdot \log_2(T/2) \cdot \log_2(N+1)$$

$$\text{if} \quad c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) - a \cdot (\log_2 T - 1) \cdot \log_2(N+1)$$

$$\text{if} \quad c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) + a \cdot \log_2(N+1) - a \cdot \log_2 T \cdot \log_2(N+1)$$

$$\text{if} \quad c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2(N+1)$$

$$\text{if} \quad \frac{c_1}{\log_2(N+1)} + a_1 \leq a$$

$$\text{if} \quad a \geq \frac{c_1}{\log_2(N+1)} + a_1$$

$$\text{if} \quad a \geq c_1 + a_1, \qquad \text{| } \log_2(N+1) \geq 1 \text{ as } N \geq 1$$

and the inequality above holds by definition of a. Let us now consider the case when T is odd. We have

$$\begin{aligned}
\text{depth}(T, N) &\leq \text{depth}(\lceil T/2 \rceil, N) + c_1 + a_1 \log_2(N+1) && \text{| Eq. (73)} \\
&\leq \text{depth}((T+1)/2, N) + c_1 + a_1 \log_2(N+1) && \text{| } \lceil T/2 \rceil = (T+1)/2 \text{ as } T \text{ is odd} \\
&\leq c + a \cdot \log_2((T+1)/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1). && \text{| Eq. (75)}
\end{aligned}$$

Thus, it suffices to show that the following inequality holds:

$$c + a \cdot \log_2((T+1)/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1) \leq c + a \cdot \log_2 T \cdot \log_2(N+1).$$

$$\begin{aligned}
& c + a \cdot \log_2((T+1)/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1) \leq c + a \cdot \log_2 T \cdot \log_2(N+1) \\
\text{if} \quad & a \cdot \log_2((T+1)/2) \cdot \log_2(N+1) + c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) \\
\text{if} \quad & c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) - a \cdot \log_2((T+1)/2) \cdot \log_2(N+1) \\
\text{if} \quad & c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) - a \cdot (\log_2(T+1) - 1) \cdot \log_2(N+1) \\
\text{if} \quad & c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2 T \cdot \log_2(N+1) + a \cdot \log_2(N+1) - a \cdot \log_2(T+1) \cdot \log_2(N+1) \\
\text{if} \quad & c_1 + a_1 \log_2(N+1) \leq a \cdot (\log_2 T + 1 - \log_2(T+1)) \cdot \log_2(N+1) \\
\text{if} \quad & c_1 + a_1 \log_2(N+1) \leq a \cdot \log_2(2T/(T+1)) \cdot \log_2(N+1) \\
\text{if} \quad & \frac{1}{\log_2(2T/(T+1))} \cdot \left(\frac{c_1}{\log_2(N+1)} + a_1 \right) \leq a \\
\text{if} \quad & a \geq \frac{1}{\log_2(2T/(T+1))} \cdot \left(\frac{c_1}{\log_2(N+1)} + a_1 \right) \\
\text{if} \quad & a \geq \frac{1}{\log_2(4/3)} \cdot \left(\frac{c_1}{\log_2(N+1)} + a_1 \right) && \text{| } 2T/(T+1) \leq 4/3 \text{ shown below and } \log_2 \text{ monotonic} \\
\text{if} \quad & a \geq \frac{1}{\log_2(4/3)} \cdot (c_1 + a_1) && \text{| } \log_2(N+1) \geq 1 \\
\text{if} \quad & a \geq 3 \cdot (c_1 + a_1), && \text{| } 1/\log_2(4/3) \leq 3
\end{aligned}$$

and the above holds by definition of a. It remains to show $2T/(T+1) \leq 4/3$. The associated equation is

$$\begin{aligned}
& 2T/(T+1) = 4/3 \\
\Leftrightarrow \quad & 2T = 4/3(T+1) \\
\Leftrightarrow \quad & 2T = 4T/3 + 4/3 \\
\Leftrightarrow \quad & 2T - 4T/3 = 4/3 \\
\Leftrightarrow \quad & (2 - 4/3)T = 4/3 \\
\Leftrightarrow \quad & (2/3)T = 4/3 \\
\Leftrightarrow \quad & T = 4/3 \cdot 3/2 \\
\Leftrightarrow \quad & T = 2,
\end{aligned}$$

and hence the inequality holds for all $T \geq 2$.

Next we analyse the work. When $T = 1$, the work is given by the work of `ParallelIdentity`$(v_{\min}, v_{\max}, N)$ which is $O(N^2)$ by Theorem 39. When $T > 1$, the work is asymptotically bounded by sum of the work of the following:

- the work of Lines 8–9, which is $O(TN^3)$ since the lines perform $m = O(T)$ calls of `ParallelComp` on inputs of dimension N (Lines 8–9), with each call having work that is $O(N^3)$ by Theorem 37;
- the work of Line 12, which is a call of `ParallelScan` on an input of length $\ell = \lceil T/2 \rceil$ and dimension N;

- the work of Lines 15–19, which is $O(TN^3)$ since the lines perform for T times either an assignment of a previously computed value (Line 17) which is $O(1)$ or a call of `ParallelComp` on an input of dimension N (Lines 19), with each call having work that is $O(N^3)$ by Theorem 37.

Thus, there exist constants a_0, a_1, c_0, c_1 such that

$$\text{work}(T,N) \leq \begin{cases} c_0 + a_0 \cdot N^2 & \text{if } T = 1, \\ \text{work}(\lceil T/2 \rceil, N) + c_1 + a_1 \cdot TN^3 & \text{otherwise.} \end{cases} \tag{76}$$

We show $\text{work}(T,N) \in O(TN^3)$ by showing that, for $c = c_0$ and $a = \max(a_0, 4(a_1 + c_1))$, the following bound holds

$$\text{work}(T,N) \leq c + a \cdot TN^3. \tag{77}$$

We proceed by induction on T. In the base case $T = 1$, and we have

$$\begin{aligned} \text{work}(T,N) &\leq c_0 + a_0 \cdot N^2 && \text{| Eq. (76)} \\ &= c + a_0 \cdot N^2 && \text{| } c = c_0 \text{ by def.} \\ &\leq c + a \cdot N^2 && \text{| } a_0 \leq a \text{ by def. of } a \\ &= c + a \cdot TN^2 && \text{| } T = 1 \\ &\leq c + a \cdot TN^3, && \text{| } N^2 \leq N^3 \end{aligned}$$

as required.

In the inductive case $T > 1$, and we assume by induction that, for all $T' \leq T - 1$,

$$\text{work}(T',N) \leq c + a \cdot T'N^3. \tag{78}$$

We consider two separate cases, according to whether T is even or odd. Let us first consider the case when T is even. We have

$$\begin{aligned} \text{work}(T,N) &\leq \text{work}(\lceil T/2 \rceil, N) + c_1 + a_1 TN^3 && \text{| Eq. (76)} \\ &= \text{work}(T/2, N) + c_1 + a_1 TN^3 && \text{| } T \text{ is even} \\ &\leq c + a(T/2)N^3 + c_1 + a_1 TN^3. && \text{| Eq. (78)} \end{aligned}$$

Thus, it suffices to show that the following inequality holds:

$$c + a(T/2)N^3 + c_1 + a_1 TN^3 \leq c + aTN^3.$$

We have

$$\begin{aligned} & c + a(T/2)N^3 + c_1 + a_1 TN^3 \leq c + aTN^3 \\ \text{if} \quad & a(T/2)N^3 + c_1 + a_1 TN^3 \leq aTN^3 \\ \text{if} \quad & a(1/2)TN^3 + c_1 + a_1 TN^3 \leq aTN^3 \\ \text{if} \quad & c_1 + a_1 TN^3 \leq aTN^3 - a(1/2)TN^3 \\ \text{if} \quad & c_1 + a_1 TN^3 \leq a(1/2)TN^3 \\ \text{if} \quad & \frac{2(c_1 + a_1 TN^3)}{TN^3} \leq a \\ \text{if} \quad & 2\left(\frac{c_1}{TN^3} + a_1\right) \leq a \\ \text{if} \quad & a \geq 2\left(\frac{c_1}{TN^3} + a_1\right) \\ \text{if} \quad & a \geq 2\left(c_1 + a_1\right) && \text{| } T, N > 1 \end{aligned}$$

and the above inequality holds by definition of a. Let us now consider the case when T is odd. We have

$$\begin{aligned} \text{work}(T,N) &\leq \text{work}(\lceil T/2 \rceil, N) + c_1 + a_1 TN^3 && \text{| Eq. (76)} \\ &= \text{work}\left((T+1)/2, N\right) + c_1 + a_1 TN^3 && \text{| } \lceil T/2 \rceil = (T+1)/2 \text{ as } T \text{ is odd} \\ &\leq c + a \cdot \left((T+1)/2\right)N^3 + c_1 + a_1 TN^3. && \text{| Eq. (78)} \end{aligned}$$

Thus, it suffices to show that the following inequality holds:

$$c + a \cdot \big((T+1)/2\big)N^3 + c_1 + a_1 T N^3 \leq c + aTN^3.$$

We have

$$\begin{aligned}
& c + a \cdot \big((T+1)/2\big)N^3 + c_1 + a_1 T N^3 \leq c + aTN^3 \\
\text{if} \quad & a \cdot \big((T+1)/2\big)N^3 + c_1 + a_1 T N^3 \leq aTN^3 \\
\text{if} \quad & c_1 + a_1 T N^3 \leq aTN^3 - a \cdot \big((T+1)/2\big)N^3 \\
\text{if} \quad & c_1 + a_1 T N^3 \leq aTN^3 - a \cdot \frac{T+1}{2T} \cdot TN^3 \\
\text{if} \quad & c_1 + a_1 T N^3 \leq a \cdot \left(1 - \frac{T+1}{2T}\right) \cdot TN^3 \\
\text{if} \quad & c_1 + a_1 T N^3 \leq a \cdot \frac{2T - T - 1}{2T} \cdot TN^3 \\
\text{if} \quad & c_1 + a_1 T N^3 \leq a \cdot \frac{T-1}{2T} \cdot TN^3 \\
\text{if} \quad & \frac{c_1 + a_1 T N^3}{TN^3} \leq a \cdot \frac{T-1}{2T} \\
\text{if} \quad & \frac{2T}{T-1} \cdot \frac{c_1 + a_1 T N^3}{TN^3} \leq a \\
\text{if} \quad & a \geq \frac{2T}{T-1} \cdot \frac{c_1 + a_1 T N^3}{TN^3} \\
\text{if} \quad & a \geq \frac{2T}{T-1} \cdot \left(\frac{c_1}{TN^3} + a_1\right) \\
\text{if} \quad & a \geq \frac{2T}{T-1} \cdot (c_1 + a_1) && \mid T, N \geq 1 \\
\text{if} \quad & a \geq 4 \cdot (c_1 + a_1), && \mid 2T/(T-1) \leq 4 \text{ shown below}
\end{aligned}$$

and the above holds by definition of a. It remains to show $2T/(T-1) \leq 4$. The associate equation is

$$\begin{aligned}
& 2T/(T-1) = 4 \\
\Leftrightarrow \quad & 2T = 4(T-1) \\
\Leftrightarrow \quad & T = 2(T-1) \\
\Leftrightarrow \quad & T = 2T - 2 \\
\Leftrightarrow \quad & 2 = T,
\end{aligned}$$

and hence the inequality holds for all $T > 1$. □

Lemma 28. *ParallelRecEval solves MinMax Recurrence Evaluation with depth $O(\log N \cdot \log T)$ and work $O(TN^3)$ for T the number of steps and N the state dimension. Thus, it runs in time $O(\log N \cdot \log T + TN^3/P)$ given P processors.*

Proof. Let (X_{init}, A, B) be an input of MinMax Recurrence Evaluation. We first show correctness and then complexity.

Correctness. Let $Y \in \mathbb{R}^{T \times N}$ be such that $Y[0] = \circledast(X_{\text{init}}, A[0], B[0])$ and $Y[t] = \circledast(Y[t-1], A[t], B[t])$ for every $t \in [1..T-1]$. It suffices to show that the variable X of the algorithm satisfies $X = Y$ when the return instruction of Line 10 is executed.

After executing Line 2, we have that $v_{\min}$ is the minimum scalar value occurring in the input, by Theorem 33. After executing Line 3, we have that $v_{\max}$ is the minimum scalar value occurring in the input, by Theorem 34. After executing Line 4, since each $\langle A[t], B[t] \rangle$ belonds to the MinMax monoid $\mathcal{M}_{[v_{\min}, v_{\max}, N}$, by Theorem 40 we have that (C, D) is such that

$$(C[t], D[t]) = (A[0], B[0]) \circledast \cdots \circledast (A[t-1], B[t-1]) \circledast \mathcal{E}, \quad \forall t \in [0..T-1], \tag{79}$$

for $\mathcal{E}$ the identity element of the MinMax monoid $\mathcal{M}_{[v_{\min},v_{\max},N}$. Let us now consider Lines 7–9 for $t \in [0..T-1]$. After executing Line 8, it holds that

$$
\begin{aligned}
&(Z[t], W[t]) && && (80)\\
&= (C[t], D[t]) \circledast (A[t], B[t]) && \text{| Line 8 and Theorem 37} && (81)\\
&= \big((A[0], B[0]) \circledast \cdots \circledast (A[t-1], B[t-1]) \circledast \mathcal{E}\big) \circledast (A[t], B[t]) && \text{| Eq. (79)} && (82)\\
&= (A[0], B[0]) \circledast \cdots \circledast (A[t-1], B[t-1]) \circledast \mathcal{E} \circledast (A[t], B[t]) && \text{| associativity} && (83)\\
&= (A[0], B[0]) \circledast \cdots \circledast (A[t-1], B[t-1]) \circledast (A[t], B[t]) && \text{| } \mathcal{E} \text{ is identity} && (84)\\
&= (A[0], B[0]) \circledast \cdots \circledast (A[t], B[t]). && && (85)
\end{aligned}
$$

After executing Line 9, by Theorem 38 it holds that

$$X[t] = (Z[t] \otimes X_{\text{init}}) \oplus W[t]. \tag{86}$$

We conclude the proof by showing that the r.h.s. above equals $Y[t]$ as required. Let us recall the default action τ of the monoid $\mathcal{M}_{[v_{\min},v_{\max}],N}$ (Definition 222), and let us introduce the notation $\tau_i(x) := \tau(A[i], B[i], x)$ for each $i \in [0..T-1]$, and then the notation $\tau_{0:i} := (\tau_0 \circ \cdots \circ \tau_i)$ for each $i \in [0..T-1]$. By the definition of the default action, we have

$$Y[t] = \tau_{0:t}(X_{\text{init}}), \tag{87}$$

and by Proposition 36 we have

$$\tau_{0:t} = \tau_{M_t} \quad \text{for } M_t := (A[0], B[0]) \circledast \cdots \circledast (A[t], B[t]). \tag{88}$$

Then,

$$
\begin{aligned}
Y[t] &= \tau_{0:t}(X_{\text{init}}) && \text{| Eq. (87)} && (89)\\
&= \tau_{M_t}(X_{\text{init}}) && \text{| Eq. (88)} && (90)\\
&= \tau(Z[t], W[t], X_{\text{init}}) && \text{| } M_t = (Z[t], W[t]) \text{ by Eq. (85)} && (91)\\
&= (Z[t] \otimes X_{\text{init}}) \oplus W[t] && \text{| def. of } \tau && (92)\\
&= X[t], && \text{| Eq. (86)} && (93)
\end{aligned}
$$

as required.

Complexity. The depth is given by the sum of:

- the sum of the depths of `ParallelMin` and `ParallelMax` (Lines 2–3) on input of length $N + TN^2 + TN$, each having depth $O(\log(N + TN^2 + TN))$ by Theorems 33,34, and hence the sum of their depths is $2 \cdot O(\log(N + TN^2 + TN)) = O(\log(TN)) = O(\log T + \log N)$;
- the depth of `ParallelScan` (Line 4) on an input having T steps and dimension N, which is $O(\log T \cdot \log N)$ by Theorem 40;
- the maximum for $t \in [0..T-1]$ of
  - the depth of `ParallelComp` (Line 8) on input of dimension N, which is $O(\log N)$ by Theorem 37;
  - the depth of `ParallelApply` (Line 9) on input of dimension N, which is $O(\log N)$ by Theorem 38.

Thus the depth is $O(\log T + \log N) + O(\log T \cdot \log N) + O(\log N)$, which is $O(\log T \cdot \log N)$ as claimed.

The work is given by the sum of:

- the sum of the work of `ParallelMin` and `ParallelMax` (Lines 2–3) on input of length $N + TN^2 + TN$, each performing work $O(N + TN^2 + TN)$ by Theorems 33,34, and hence their total work is $2 \cdot O(N + TN^2 + TN) = O(TN^2)$;

- the work of `ParallelScan` (Line 4) on an input having T steps and dimension N, which is $O(TN^3)$ by Theorem 40;
- the sum of the work of the following for $t \in [0..T-1]$:
  - the work of `ParallelComp` (Line 8) on input of dimension N, which is $O(N^3)$ by Theorem 37;
  - the work of `ParallelApply` (Line 9) on input of dimension N, which is $O(N^2)$ by Theorem 38.

Thus the work is $O(TN^2) + O(TN^3) + T \cdot (O(N^3) + O(N^2))$, which is $O(TN^3)$ as claimed. □

Theorem 41. *MinMax Recurrence Evaluation can be solved in parallel with depth $O(\log N \cdot \log T)$ and work $O(TN^3)$ for T the number of steps and N the state dimension. Thus, it can be solved in time $O(\log N \cdot \log T + TN^3/P)$ given P processors, and hence in $O(\log N \cdot \log T)$ if $P \in \Omega(TN^3)$.*

Proof. Immediate by Lemma 28. □

Lemma 29. *On input $\langle W, U\rangle$ for $W \in \mathbf{W}_{\mathbf{s}}^{\text{layer}}$ with size $\mathbf{s} = \langle d_{\text{in}}, d_{\text{mlp}}, n_{\text{mlp}}, d_{\text{out}}, n_{\text{units}}, d_{\text{state}}\rangle$, and $U \in \mathbb{R}^{T \times d_{\text{in}}}$, the output of ParLayerEval is $S(U)$ for $S = \text{Layer}(W)$, its depth is*

$$O\Big(\log T \cdot \log d_{\text{state}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{in}})\Big)$$

and its work is

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{in}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{out}} + n_{\text{units}} d_{\text{state}})\big)\Big)$$

Proof. We note that the algorithm is obtained from the sequential algorithm SeqLayerEval in a straighforward manner. Thus, correctness follows by the same arguments as in the proof of Lemma 26. In particular, `SeqRecEval` is now replaced by `ParRecEval`, which yields the same result, by Lemma ParallelRecEval; and the loops in `SeqLayerEval` have independent iterations and hence their iterations can be executed in parallel.

Next we analyse the complexity. The depth is asymptotically bounded by the sum of the following depths:

- Line 8: the depth of `EvaluateMLP`, where the MLP has input dimension d_{in}, output dimension d_{state}^2, and n_{mlp} hidden layers of dimension $d_{\text{mlp,u}} := \lceil d_{\text{mlp}}/n_{\text{units}} \rceil$; its depth is $O(\log d_{\text{in}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}})$;
- Line 9: the depth of `EvaluateMLP` where the MLP has input dimension d_{in}, output dimension d_{state}, and n_{mlp} hidden layers of dimension $d_{\text{mlp,u}} := \lceil d_{\text{mlp}}/n_{\text{units}} \rceil$; its depth is $O(\log d_{\text{in}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}})$;
- Line 13: the depth SeqRecEval on inputs of dimension d_{state} and length T; its depth is $O(\log T \cdot \log d_{\text{state}})$ by Theorem 28;
- Line 16: the depth of `EvaluateMLP`, where the MLP has input dimension $2n_{\text{units}} d_{\text{state}} + d_{\text{in}}$, output dimension d_{out}, and n_{mlp} hidden layers of dimension d_{mlp}; its depth is $O(\log(n_{\text{units}} d_{\text{state}} + d_{\text{in}}) + n_{\text{mlp}} \cdot \log d_{\text{mlp}})$;

Thus, the depth is

$$\begin{aligned}
&O(\log d_{\text{in}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}}) + O(\log T \cdot \log d_{\text{state}}) + O(\log(n_{\text{units}} d_{\text{state}} + d_{\text{in}}) + n_{\text{mlp}} \cdot \log d_{\text{mlp}}) \\
&= O\Big(\log d_{\text{in}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log T \cdot \log d_{\text{state}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{in}}) + n_{\text{mlp}} \cdot \log d_{\text{mlp}}\Big) \\
&= O\Big(n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log T \cdot \log d_{\text{state}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{in}})\Big)
\end{aligned}$$

The work is asymptotically bounded by the sum of the following work:

- Line 8: the work of $O(n_{\text{units}} \cdot T)$ executions of `EvaluateMLP`, where the MLP has input dimension d_{in}, output dimension d_{state}^2, and n_{mlp} hidden layers of dimension $d_{\text{mlp,u}} := \lceil d_{\text{mlp}}/n_{\text{units}} \rceil$; the time of each evaluation is $O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,i}} d_{\text{state}}^2)$;

- Line 9: the work of $O(n_{\text{units}} \cdot T)$ executions of `EvaluateMLP` where the MLP has input dimension d_{in}, output dimension d_{state}, and n_{mlp} hidden layers of dimension $d_{\text{mlp,u}} := \lceil d_{\text{mlp}}/n_{\text{units}} \rceil$; the time of each evaluation is $O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}})$;
- Line 13: the work of $O(n_{\text{units}})$ executions of SeqRecEval on inputs of dimension d_{state} and length T; its execution time is $O(T d_{\text{state}}^3)$ by Theorem 25;
- Line 16: the work of $O(T)$ executions of `EvaluateMLP`, where the MLP has input dimension $2n_{\text{units}} d_{\text{state}} + d_{\text{in}}$, output dimension d_{out}, and n_{mlp} hidden layers of dimension d_{mlp}; the time of each evaluation is $O\big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)$.

Thus, the work is

$$
\begin{aligned}
&O(n_{\text{units}} \cdot T) \cdot O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}^2) + \\
&O(n_{\text{units}} \cdot T) \cdot O(d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}) + \\
&O(n_{\text{units}}) \cdot O(T d_{\text{state}}^3) + \\
&O(T) \cdot O\big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp}} d_{\text{out}}\big) \\
&= \\
&O\Big(n_{\text{units}} \cdot T \cdot (d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}^2)\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot (d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}})\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot d_{\text{state}}^3\Big) + \\
&O\Big(T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(n_{\text{units}} \cdot T \cdot (d_{\text{in}} d_{\text{mlp,u}} + n_{\text{mlp}} d_{\text{mlp,u}}^2 + d_{\text{mlp,u}} d_{\text{state}}^2)\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot d_{\text{state}}^3\Big) + \\
&O\Big(T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot (d_{\text{in}} d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{state}}^2)\Big) + \\
&O\Big(n_{\text{units}} \cdot T \cdot d_{\text{state}}^3\Big) + \\
&O\Big(T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot (d_{\text{in}} d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{state}}^2) + \\
&\quad n_{\text{units}} \cdot T \cdot d_{\text{state}}^3 + \\
&\quad T \cdot \big((n_{\text{units}} d_{\text{state}} + d_{\text{in}}) \cdot d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot \big(d_{\text{in}} d_{\text{mlp}} + n_{\text{mlp}} d_{\text{mlp}}^2 + d_{\text{mlp}} d_{\text{state}}^2 + n_{\text{units}} \cdot d_{\text{state}}^3 + n_{\text{units}} \cdot d_{\text{mlp}} d_{\text{state}} + d_{\text{mlp}} d_{\text{out}}\big)\Big) \\
&= \\
&O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{in}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{out}} + n_{\text{units}} d_{\text{state}})\big)\Big)
\end{aligned}
$$

□

Theorem 42. *On input $\langle W, U \rangle$ for $W \in \mathbf{W}^{\text{rnc}}_{\mathbf{s}}$ with size*

$$\mathbf{s} = \langle d_{\text{in}}, d_{\text{out}}, d_{\text{model}}, d_{\text{state}}, n_{\text{units}}, n_{\text{layers}}, d_{\text{mlp}}, n_{\text{mlp}} \rangle,$$

and $U \in \mathbb{R}^{T \times d_{\text{in}}}$, the output of ParCascEval is $S(U)$ for $S = \text{RNC}(W)$, its depth is

$$O\Big(n_{\text{layer}} \cdot \big(\log T \cdot \log d_{\text{state}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{in}} + d_{\text{model}})\big)\Big),$$

and the work is

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{in}} + + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{model}} + d_{\text{out}} + n_{\text{units}} d_{\text{state}})\big)\Big).$$

Furthermore, for fixed n_{mlp} and $d := \max\{d_{\text{in}}, d_{\text{out}}, d_{\text{mlp}}, n_{\text{units}} \cdot d_{\text{state}}\}$, the depth is

$$O\Big(n_{\text{layer}} \cdot \big(\log T \cdot \log d_{\text{state}} + \log d\big)\Big),$$

and the work is

$$O\Big(T \cdot n_{\text{layers}} \cdot d \cdot \big(d + d_{\text{state}}^2\big)\Big).$$

Proof. We note that the algorithm is obtained from the sequential algorithm SeqCascEval in a straighforward manner. Thus, correctness follows by the same arguments as in the proof of Lemma 27. In particular, `SeqLayerEval` is now replaced by `ParLayerEval`, which yields the same result, by Lemma 29; and the loops in `SeqCascEval` have independent iterations and hence their iterations can be executed in parallel.

Then, the depth and work are given by the sum of depth and work, respectively, of the calls to `ParLayerEval`, to evaluate a layer having input dimension d_{in} and output dimension d_{model} for $i = 0$, input and output dimension d_{model} for each $i \in [1..n_{\text{layers}} - 2]$, and input dimension d_{model} and output dimension d_{out} for $i = n_{\text{layers}} - 1$. For the rest, the size of each layer is described by $\langle d_{\text{state}}, n_{\text{units}}, d_{\text{mlp}}, n_{\text{mlp}} \rangle$. By Lemma 29 the depth of the execution for the first layer is

$$O\Big(\log T \cdot \log d_{\text{state}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{in}})\Big)$$

the depth for each layer $i \in [1..n_{\text{layers}} - 1]$ is

$$O\Big(\log T \cdot \log d_{\text{state}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{model}})\Big)$$

Thus, overall the depth is

$$O\Big(n_{\text{layer}} \cdot \big(\log T \cdot \log d_{\text{state}} + n_{\text{mlp}} \cdot \log d_{\text{mlp}} + \log(n_{\text{units}} d_{\text{state}} + d_{\text{in}} + d_{\text{model}})\big)\Big).$$

For fixed n_{mlp} and $d := \max\{d_{\text{in}}, d_{\text{out}}, d_{\text{mlp}}, n_{\text{units}} \cdot d_{\text{state}}\}$,

$$O\Big(n_{\text{layer}} \cdot \big(\log T \cdot \log d_{\text{state}} + \log d\big)\Big).$$

By Lemma 29 the work of the execution for the first layer is

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{in}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{model}} + n_{\text{units}} d_{\text{state}})\big)\Big)$$

the work for each layer $i \in [1..n_{\text{layers}} - 2]$ is

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{model}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{model}} + n_{\text{units}} d_{\text{state}})\big)\Big)$$

and for the last layer, when $i = n_{\text{layers}} - 1$,

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{model}} + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{out}} + n_{\text{units}} d_{\text{state}})\big)\Big)$$

Thus, overall the work is

$$O\Big(T \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d_{\text{mlp}} \cdot (d_{\text{in}} + + n_{\text{mlp}} d_{\text{mlp}} + d_{\text{state}}^2 + d_{\text{model}} + d_{\text{out}} + n_{\text{units}} d_{\text{state}})\big)\Big)$$

For fixed n_{mlp} and $d := \max\{d_{\text{in}}, d_{\text{out}}, d_{\text{mlp}}, n_{\text{units}} \cdot d_{\text{state}}\}$,

$$\begin{aligned}
&O\Big(T \cdot n_{\text{layers}} \cdot \big(n_{\text{units}} \cdot d_{\text{state}}^3 + d \cdot (d + d_{\text{state}}^2)\big)\Big) \\
&= O\Big(T \cdot n_{\text{layers}} \cdot \big(d \cdot d_{\text{state}}^2 + d \cdot (d + d_{\text{state}}^2)\big)\Big) \\
&= O\Big(T \cdot n_{\text{layers}} \cdot d \cdot \big(d + d_{\text{state}}^2\big)\Big).
\end{aligned}$$

□

L BIBS and BIBO Stability of MinMax RNCs

Table of Contents

L.1 A one-step MinMax state-value bound 189

L.2 Uniform-BIBS of MinMax recurrent units and of MinMax RNCs 189

L.3 BIBS and BIBO of MinMax RNCs 190

This appendix proves the stability result stated as Theorem 7. More generally, we establish *uniform bounded-input bounded-state (uBIBS)* and *uniform bounded-input bounded-output (uBIBO)* properties for parametric MinMax RNCs, together with their weaker variants, as defined in Appendix F.

The key observation is that a MinMax update

$$x_t \;=\; x_{t-1} \circledast (R_t, s_t) \;=\; (R_t \otimes x_{t-1}) \oplus s_t$$

satisfies a one-step bound depending only on the current reset and set values, and not on the magnitude of the previous state. Thus, whenever the reset and set functions are bounded on bounded input and parameter sets, the state remains uniformly bounded independently of the time horizon.

We first formalise this observation and use it to prove uBIBS for individual MinMax recurrent units. We then lift the property to arbitrary MinMax RNC dynamics using the general closure results for recurrent neural circuits. Finally, boundedness of the analytic output map yields uBIBO, from which the corresponding weak BIBS/BIBO properties follow. Theorem 7 is then obtained directly from these stronger uniform results.

L.1 A one-step MinMax state-value bound

The following elementary bound is used to establish uBIBS of MinMax recurrent units.

Lemma 30 (Value bound for a MinMax update)**.** *Let $x \in \mathbb{R}^n$, let $R \in \mathbb{R}^{n\times n}$, let $s \in \mathbb{R}^n$, and let $F := x \circledast (R, s)$. Then*

$$\|F\|_\infty \le \max\left\{\|s\|_\infty, \|R\|_{\max}\right\}.$$

Proof. Let $M := \max\left\{\|s\|_\infty, \|R\|_{\max}\right\}$. We have

$$(F)_i \;=\; \max\left\{(s)_i,\; \max_{j\in[1..n]} \min\left((R)_{i,j}, (x)_j\right)\right\} \qquad \forall i \in [1..n].$$

Since

$$\min\left((R)_{i,j}, (x)_j\right) \;\le\; (R)_{i,j} \;\le\; M \qquad \text{and} \qquad (s)_i \;\le\; M \qquad \forall i, j \in [1..n],$$

we have

$$(F)_i \le M \qquad \forall i \in [1..n].$$

Furthermore, since $(F)_i$ is a maximum including $(s)_i$ and $|(s)_i| \le \|s\|_\infty \le M$, we have

$$(F)_i \;\ge\; (s)_i \;\ge\; -M \qquad \forall i \in [1..n].$$

Hence

$$|(F)_i| \le M \qquad \forall i \in [1..n],$$

proving the claim. □

L.2 Uniform-BIBS of MinMax recurrent units and of MinMax RNCs

Proposition 37 (uBIBS of parametric MinMax neurons)**.** *The class $\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]$ satisfies uBIBS.*

Proof. Let us consider an arbitrary $r \in \mathbb{R}^{\ge 0}$ and an arbitrary parametric MinMax neuron $D \in \mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]$ of state dimension n given by

$$D = \langle X, U, f; \Theta \,|\, R, s\rangle.$$

By Definition 201, the joint reset–set function $\rho := (R, s)$ is a parametric analytic neural map. Therefore, Corollary 14 yields $M_r \in \mathbb{R}^{\geq 0}$ such that

$$\|\rho(u;\theta)\| \leq M_r \qquad \forall u \in U_{\leq r},\ \forall \theta \in \Theta_{\leq r}.$$

Since $\|R(u;\theta)\|_{\max} \leq \|\rho(u;\theta)\|$ and $\|s(u;\theta)\|_\infty \leq \|\rho(u;\theta)\|$, it follows that

$$\max\left\{\|R(u;\theta)\|_{\max}, \|s(u;\theta)\|_\infty\right\} \leq M_r.$$

Let us fix arbitrary $x_0 \in X_{\leq r}$, $u \in U^\omega_{\leq r}$, and $\theta \in \Theta_{\leq r}$. Let

$$R_t := R(u_t;\theta), \qquad s_t := s(u_t;\theta). \qquad \forall t \in \mathbb{T}.$$

By (30),

$$\|x_t\|_\infty \leq M_r \qquad \forall t \in \mathbb{T}.$$

Hence, using $\|x\| \leq \sqrt{n} \cdot \|x\|_\infty$ and the assumption $\|x_0\| \leq r$,

$$\sup_{t\in\mathbb{N}} \|x_t\| \leq \max\left\{r,\ \sqrt{n} \cdot M_r\right\}.$$

Since x_0, u, and θ are arbitrary, this bound is uniform over the bounded input sequence, initial state, and parameter value. Thus D satisfies uBIBS. Since D is arbitrary in $\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]$, we conclude that the class $\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]$ satisfies uBIBS. □

Corollary 20 (uBIBS of parametric MinMax RNC dynamics)**.** *The class* $\mathrm{RNC}_{\mathrm{dyn}}\left(\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]\right)$ *satisfies uBIBS.*

Proof. By Proposition 37, the class $\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]$ satisfies uBIBS. The claim follows from Theorem 20. □

L.3 BIBS and BIBO of MinMax RNCs

Since uBIBS implies wBIBS, Proposition 37 and Corollary 20 also yield wBIBS for parametric MinMax neurons and MinMax RNC dynamics.

Theorem 43 (uBIBS and uBIBO of parametric MinMax RNCs)**.** *The class* $\mathrm{RNC}\left(\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]\right)$ *satisfies uBIBS and uBIBO, and hence also wBIBS and wBIBO.*

Proof. Let $S \in \mathrm{RNC}\left(\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]\right)$ be arbitrary. There exist $D \in \mathrm{RNC}_{\mathrm{dyn}}\left(\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]\right)$, an initial state x_0 of D, and a parametric analytic map h such that

$$S = (D, x_0) \triangleright h.$$

By Corollary 20, D satisfies uBIBS. Hence, by Definition 179, S satisfies uBIBS and therefore also wBIBS.

It remains to prove uBIBO. Fix $r \in \mathbb{R}^{\geq 0}$ and set

$$\hat{r} := \max\{r, \|x_0\|\}.$$

Since D satisfies uBIBS, there exists $B^X_{\hat{r}} \in \mathbb{R}^{\geq 0}$ such that, for every

$$\theta \in \Theta_{\leq r}, \qquad u \in U^\omega_{\leq r},$$

the state sequence $(x_t)_{t\in\mathbb{T}}$ of $D[\theta]$ starting from the fixed state x_0 satisfies

$$\sup_{t\in\mathbb{T}} \|x_t\| \leq B^X_{\hat{r}}.$$

Indeed, $\theta \in \Theta_{\leq \hat{r}}$, $u \in U^\omega_{\leq \hat{r}}$, and $x_0 \in X_{\leq \hat{r}}$.

Consequently, the output contexts

$$p_t := (x_{t-1}, u_t, x_t) \qquad \forall t \in \mathbb{N}^{>0}$$

are uniformly bounded. Hence there exists $\rho_r \in \mathbb{R}^{\geq 0}$, depending only on r, x_0, and $B^X_{\hat{r}}$, such that

$$\|p_t\| \leq \rho_r \qquad \forall t \in \mathbb{N}^{>0}.$$

Let θ_h be the parameter block of the output map. Since $\|\theta_h\| \leq \|\theta\| \leq r$, setting

$$\bar{r} := \max\{r, \rho_r\},$$

Corollary 14 applied to h yields $B_r^Y \in \mathbb{R}^{\geq 0}$ such that

$$\|y_t\| = \|h(p_t; \theta_h)\| \leq B_r^Y \qquad \forall t \in \mathbb{N}^{>0},$$

uniformly over all parameter values and input sequences bounded by r. Thus S satisfies uBIBO, and therefore also wBIBO. Since S was arbitrary, the claim follows. □

We restate Theorem 7 equivalently—but more precisely—in the notation of Appendix F as follows.

Theorem 44 (Equivalent formulation of Theorem 7)**.** *Let $S = \langle X, U, f, x_0, Y, h; \Theta \rangle$ be a parametric MinMax RNC. For every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that, for every*

$$\theta \in \Theta_{\leq r}, \qquad u = (u_t)_{t=1}^T \in U_{\leq r}^+,$$

the corresponding state sequence $(x_t)_{t=1}^T$ and output sequence $(y_t)_{t=1}^T$, given by

$$x_t := \mathrm{dyn}(\check{\mathrm{S}}[\theta])(x_0, u_{[1..t]}), \qquad y_t := \check{S}[\theta](u_{[1..t]}) \qquad \forall t \in [1..T],$$

satisfy

$$\max_{t \in [1..T]} \|x_t\| \leq B_r, \qquad \max_{t \in [1..T]} \|y_t\| \leq B_r.$$

Proof. By Theorem 43, there exist uniform state and output bounds $B_r^X, B_r^Y \in \mathbb{R}^{\geq 0}$. The claim follows with $B_r := \max\{B_r^X, B_r^Y\}$. □

M Gradients of MinMax RNCs

Table of Contents

M.1 Preliminaries . . . 192
M.2 Relevant Class of Functions . . . 192
M.3 Pseudo-differentiating Min and Max Functions . . . 193
M.4 Analytic-MinMax Computation Graphs . . . 197
M.5 Virtual Graph-Extension Notation for Analytic-Minax Graphs . . . 199
M.6 Properties of the Analytic-MinMax Pseudo-Derivative Operator . . . 202
M.7 MinMax Recurrence Computation Graphs . . . 204
M.8 MinMaxRNC State Computation Graphs . . . 206
M.9 Sequential Computation Graphs for MinMax RNCs . . . 210
M.10 Parallel-Scan Computation Graphs for MinMax RNCs . . . 212
M.11 BIBSD: Uniform Selected uBIBSD for MinMax RNCs . . . 219
M.12 BIBLG: Bounded Loss Gradients of MinMax RNCs . . . 222
M.13 Non-necessarily Vanishing State-to-State Derivatives . . . 227
M.14 Deferred Proofs for MinMax Pseudo-Derivatives . . . 231

This appendix provides the full proofs of the gradient results stated in Section 6. The results have two complementary goals.

First, we prove Theorem 8, showing that the derivatives relevant to training parametric MinMax RNCs remain uniformly bounded on bounded sets, independently of the input horizon. More precisely, we show that, for every admissible MinMax-algebraic computation graph selected for the loss, the corresponding MinMax selected pseudo-derivative is uniformly bounded; under the appropriate intensional-admissibility assumptions, it is an intensional derivative, and the same bound consequently holds for the ordinary loss derivative almost everywhere.

The main difficulty is that a MinMax recurrence may be represented by different computation-graph topologies. In particular, sequential, balanced-tree, parallel-scan, and mixed evaluation strategies may induce different selected derivatives at nondifferentiable points. We therefore prove a stronger, topology-independent result. We first show that selected pseudo-derivatives of scalar minimum and maximum nodes are non-expansive in a suitable row norm. We then introduce MinMax-algebraic recurrence computation graphs and prove that the selected derivative of any such graph is controlled only by the selected derivatives of its reset and set source graphs, independently of both the recurrence horizon and the internal graph topology. Combining this property with uniform derivative bounds for the analytic parts of the architecture yields graph-uniform selected uBIBSD for complete MinMax RNC dynamics. Finally, we propagate this state-derivative bound through the output and loss maps to obtain the bounded-loss-derivative statement of Theorem 8.

Second, we prove Theorem 9. which show that bounded state derivatives need not vanish with the sequence length.

The appendix is therefore organised around these two aspects of gradient behaviour: uniform boundedness over arbitrary horizons and admissible computation-graph topologies, and the existence of MinMax recurrences whose state derivatives remain non-vanishing over arbitrarily long finite horizons.

M.1 Preliminaries

M.2 Relevant Class of Functions

Definition 223 (Analytic-MinMax primitive functions)**.** The class $\mathcal{P}_{\circledast}$ of *analytic-MinMax primitive functions* consists of:

- all analytic Euclidean functions;
- all MinMax reductions min_n and max_n for $n \in \mathbb{N}$ with $n \geq 2$ as per Definition 132.

End of Definition 223.

Definition 224 (Analytic-MinMax functions)**.** The class $\mathcal{F}_{\circledast}$ of *analytic-MinMax functions (AMM-functions)* is the closure of $\mathcal{P}_{\circledast}$.[11] Namely, it is the smallest class such that: (i) it contains $\mathcal{P}_{\circledast}$, (ii) it is closed under function tupling, and (iii) it is closed under function composition. *End of Definition 224.*

Remark 25 (Terminology). The adjective *analytic-MinMax* refers to the fact that the class includes two kinds of primitive functions:

- analytic Euclidean functions;
- MinMax reductions.

In particular, *analytic-MinMax* does not mean that every analytic-MinMax function is analytic. In fact, Min and Max reductions are generally non-differentiable at points where an extremum is attained by more than one argument. *End of Remark 25.*

We introduce the notion of *active-variable index function*, which we will use to define pseudo-derivatives of MinMax functions.

Definition 225. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a MinMax reduction. The *active-variable index function* of f is the function $I : \mathbb{R}^n \to \mathcal{P}_+([1..n])$ given by

$$I(x_1, \ldots, x_n) := \big\{ i \in [1..n] \mid x_i = f(x_1, \ldots, x_n) \big\} \qquad \forall (x_1, \ldots, x_n) \in \mathbb{R}^n.$$

M.3 Pseudo-differentiating Min and Max Functions

The derivative of min and max functions is is not guaranteed to exist everywhere.

To provide some intuition, we discuss the case of the max function $\max_2 : \mathbb{R}^2 \to \mathbb{R}$ given by $\max_2(x_1, x_2) = x_1 \oplus x_2$. Its gradient is well-defined at $(\bar{x}_1, \bar{x}_2)$ when $\bar{x}_1 \neq \bar{x}_2$, since in this case its change is given by the change of $\bar{x}_1$ if $\bar{x}_1 > \bar{x}_2$, and by the change of $\bar{x}_2$ if $\bar{x}_2 > \bar{x}_1$. However, when $\bar{x}_1 = \bar{x}_2$ the gradient of $\max_2$ at $(\bar{x}_1, \bar{x}_2)$ does not exist.

To describe the change at all points, one can employ a *subgradient* of $\max_2$, a generalisation of the gradient for convex functions. In fact, the subgradient is not unique, and one refers to the set of all subgradients as the *subdifferential*. One subgradient of $\max_2$ at $(\bar{x}_1, \bar{x}_2)$ when $\bar{x}_1 = \bar{x}_2$ is given by $(1/2, 1/2)$, which essentially says that the change of $\max_2$ is the average change of its inputs. This is in fact the subgradient of minimum (Euclidean) norm, which is the one employed in [34].

The issue with subgradients is that they are only defined for convex functions. For concave functions such as min, one can employ the dual notion of supergradient. However, compositions of min and max functions will not be convex nor concave, nor compositions that will additionally include other functions. In particular, the loss functions of MinMax RNCs are not convex nor concave in general.

To address the issue, we focus on the strategy adopted in [34] to be able to always compute a *pseudo-gradient*, with two main benefits. First, we are able to prove properties of the actual gradient, holding almost everywhere, such as its existence and boundedness. Second, we are able to prove properties for the pseudo-gradient computed by [34]; notably, the fact that it coincides with the gradient almost everywhere, and that it is always bounded, even at points where the actual gradient does not exist.

The strategy adopted in [34] is to compute a pseudo-gradient over a computation graph via the chain rule, computing locally at each node:

- the gradient of the function for the node when such function is differentiable,
- the *subgradient* of minimum norm in the case of a convex function, and
- the *supergradient* of minimum norm in the case of a concave function.

[11] The notion of *closure* is introduced in Definition 11.

The result of such computation is not guaranteed to be a gradient. Furthermore, the outcome may be different for different computation graphs—even though such computation graphs are equivalent in the sense that they represent the same function. Formally, this corresponds to the graph-induced pseudo-derivative of Definition 163.

We employ the theory of *piecewise analytic under analytic partition (PAP)* functions [43], to show that the loss functions of MinMax RNCs are differentiable almost everywhere, and that the pseudo-gradients computed in the above manner agree with the gradient almost everywhere. In particular, we show that the loss functions of MinMax RNCs are PAP; and also that, for all *algebraic computation graphs*, the corresponding pseudo-gradient is an *intensional derivative* as defined by [43].

We introduce subgradients and supergradients, cf. [42].

Definition 226 (Subgradient)**.** Let $X = \mathbb{R}^n$, let $f : X \to \mathbb{R}$ be a convex function, and let $x \in X$. A vector $g \in \mathbb{R}^n$ is a *subgradient* of f at x if, for all $z \in X$,

$$f(z) \geq f(x) + g^\top (z - x).$$

Then, the *subdifferential* $\partial_\circ f(x)$ is the set consisting of each subgradient of f at $x \in X$.

Definition 227 (Supergradient)**.** Let $X = \mathbb{R}^n$, let $f : X \to \mathbb{R}$ be a concave function, and let $x \in X$. A vector $g \in \mathbb{R}^n$ is a *supergradient* of f at x if $-g \in \partial_\circ(-f)(x)$. Then, the *superdifferential* $\partial^+_\circ f(x)$ is the set consisting of each supergradient of f at $x \in X$.

Proposition 38. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function $f(x) = \bigoplus_{i=1}^n x_i$, let I be the active-variable index function of f (Definition 225), and let $x \in \mathbb{R}^n$. Then, the subdifferential of f at x is given by*

$$\partial_\circ f(x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \textstyle\bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in I(x)} g_i = 1 \right\}.$$

Furthermore, the unique subgradient $g^\star \in \partial_\circ f(x)$ of minimum (Euclidean) norm is given by

$$g^\star = (g^\star_1, \ldots, g^\star_n) \qquad \text{where } \; g^\star_i = \begin{cases} 1/|I(x)| & \text{if } i \in I(x), \\ 0 & \text{otherwise}, \end{cases}$$

and its norm is $\|g^\star\| = 1/\sqrt{|I(x)|}$.

Proof. Deferred to Section M.14. □

Proposition 39. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function $f(x) = \bigodot_{i=1}^n x_i$, let I be the active-variable index function of f (Definition 225), and let $x \in \mathbb{R}^n$. Then, the superdifferential of f at x is given by*

$$\partial^+_\circ f(x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \textstyle\bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in I(x)} g_i = 1 \right\}.$$

Furthermore, the unique supergradient $g^\star \in \partial_\circ f(x)$ of minimum (Euclidean) norm is given by

$$g^\star = (g^\star_1, \ldots, g^\star_n) \qquad \text{where } \; g^\star_i = \begin{cases} 1/|I(x)| & \text{if } i \in I(x), \\ 0 & \text{otherwise}, \end{cases}$$

and its norm is $\|g^\star\| = 1/\sqrt{|I(x)|}$.

Proof. Deferred to Section M.14. □

Definition 228 (Analytic-MinMax pseudo-derivative operator)**.** The *analytic-MinMax pseudo-derivative operator*, abbreviated as *AMM-PD-operator*, is the pseudo-derivative operator $\tilde{\mathsf{D}}_\circledast$ applicable to $\mathcal{P}_\circledast$ such that:

- when f is analytic, we have $\tilde{\mathsf{D}}_\circledast f := \mathsf{D} f$;
- when $f : \mathbb{R}^n \to \mathbb{R}$ is the Max reduction

$$f(x) = \bigoplus_{i=1}^n x_i,$$

the pseudo-derivative $\tilde{\mathsf{D}}_\circledast f(x)$ is the unique subgradient of f at x having minimum Euclidean norm;

- when $f : \mathbb{R}^n \to \mathbb{R}$ is the Min reduction

$$f(x) = \bigodot_{i=1}^{n} x_i,$$

the pseudo-derivative $\tilde{\mathsf{D}}_{\circledast} f(x)$ is the unique supergradient of f at x having minimum Euclidean norm.

End of Definition 228.

Proposition 40. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function $f(x) = \bigoplus_{i=1}^{n} x_i$, and let I be its active variable index function (Definition 225). Let $\{I_1, \ldots, I_k\} = \mathrm{Im}(I)$, and let $A = \{A_1, \ldots, A_k\}$ where $A_i = \{x \in \mathbb{R}^n \mid I(x) = I_i\}$. Then A is a partition of $\mathbb{R}^n$.*

Proof. First, we show that each A_i is non-empty. Since $I_i \in \mathrm{Im}(I)$, there exists $x \in \mathbb{R}^n$ such that $I(x) = I_i$, hence $x \in A_i$, and hence $A_i \neq \emptyset$.

Second, we show $\bigcup_{i=1}^{k} A_i = \mathbb{R}^n$. Let $x \in \mathbb{R}^n$. Since $I(x) \in \mathrm{Im}(I)$, there exists $i \in [1..k]$ such that

$$I(x) = I_i.$$

Thus $x \in A_i \subseteq \bigcup_{i=1}^{k} A_i$.

Third, we show that the sets of A are pair-wise disjoint. Suppose that $x \in A_i \cap A_j$. Then

$$I(x) = I_i, \qquad I(x) = I_j,$$

and hence

$$I_i = I_j.$$

Since $I_1, \ldots, I_k$ are distinct, it follows that $i = j$, and hence $A_i = A_j$.

Thus the sets of A are non-empty, pair-wise disjoint, and their union is $\mathbb{R}^n$. Therefore $\mathcal{A}$ is a partition of $\mathbb{R}^n$. □

Required background. See §A.12 for the definition of *piecewise analytic under analytic partition (PAP) function* and *intensional derivatives* of a PAP function.

Proposition 41. *The function $f(x) = \bigoplus_{i=1}^{n} x_i$ is PAP, and $\tilde{\mathsf{D}}_{\circledast} f$ is an intensional derivative of f.*

Proof. We show a PAP representation γ of f. Let I be the active-variable index function of f, and let $\{I_1, \ldots, I_k\} = \mathcal{P}_+([1..n])$. For each $i \in [1..k]$, let ρ_i be an arbitrary index in I_i. For each $i \in [1..k]$ and each $j \in [1..n]$, let us define the functions $g^+_{i,j}, g^-_{i,j} : \mathbb{R}^n \to \mathbb{R}$ as

$$g^+_{i,j}(x) := \begin{cases} 1 & \text{if } j \in I_i, \\ x_{\rho_i} - x_j & \text{otherwise,} \end{cases} \qquad g^-_{i,j}(x) := \begin{cases} (x_{\rho_i} - x_j)^2 & \text{if } j \in I_i, \\ 0 & \text{otherwise.} \end{cases}$$

For each $i \in [1..k]$, Then, we define $\gamma := \{\langle A_i, f_i \rangle \mid i \in [1..k]\}$ where

$$f_i(x) := \frac{1}{|I_i|} \sum_{j \in I_i} x_j, \qquad A_i := \left\{ x \in \mathbb{R}^n \;\middle|\; \textstyle\bigwedge_{j=1}^{n} g^+_{i,j}(x) > 0 \wedge \bigwedge_{j=1}^{n} g^-_{i,j}(x) \leq 0 \right\}.$$

We first show that $\{A_i \mid i \in [1..k]\}$ is a partition of $\mathbb{R}^n$. By Proposition 40, it suffices to show that each A_i is given by $A_i = \{x \in \mathbb{R}^n \mid I(x) = I_i\}$. Namely, it suffices to show that the following equality holds for every $i \in [1..k]$,

$$\left\{ x \in \mathbb{R}^n \;\middle|\; \textstyle\bigwedge_{j=1}^{n} g^+_{i,j}(x) > 0 \wedge \bigwedge_{j=1}^{n} g^-_{i,j}(x) \leq 0 \right\} = \{x \in \mathbb{R}^n \mid I(x) = I_i\}. \tag{94}$$

For every $i \in [1..k]$, and every $x \in \mathbb{R}^n$, we have

$$\begin{aligned}
&\bigwedge_{j=1}^{n} g^+_{i,j}(x) > 0 \\
&\Leftrightarrow \bigwedge_{j \in I_i} 1 > 0 \wedge \bigwedge_{j \notin I_i} x_{\rho_i} - x_j > 0 \\
&\Leftrightarrow \bigwedge_{j \notin I_i} x_{\rho_i} > x_j
\end{aligned}$$

and

$$\begin{aligned}
&\bigwedge_{j=1}^{n} g^-_{i,j}(x) \leq 0 \\
&\Leftrightarrow \bigwedge_{j \in I_i} (x_{\rho_i} - x_j)^2 \leq 0 \wedge \bigwedge_{j \notin I_i} 0 \leq 0 \\
&\Leftrightarrow \bigwedge_{j \in I_i} (x_{\rho_i} - x_j)^2 \leq 0 \\
&\Leftrightarrow \bigwedge_{j \in I_i} x_{\rho_i} = x_j
\end{aligned}$$

and hence overall

$$\bigwedge_{j=1}^{n} g^+_{i,j}(x) > 0 \wedge \bigwedge_{j=1}^{n} g^-_{i,j}(x) \leq 0 \quad \Leftrightarrow \quad \bigwedge_{j \notin I_i} x_{\rho_i} > x_j \wedge \bigwedge_{j \in I_i} x_{\rho_i} = x_j \quad \Leftrightarrow \quad I_i = I(x).$$

Therefore, for every $i \in [1..k]$, the characterisations of the two sets in (94) are equivalent, and hence the two sets are equal.

Next, we show that γ is a representation of f; namely, we show that, for every $x \in \mathbb{R}^n$, it holds that $f_i(x) = f$ where i is the (unique) index such that $x \in A_i$. Let $x \in \mathbb{R}^n$ and let i be its partition index. We have that $f(x) = x_j$ for all $j \in I(x)$. We have shown above that $I_i = I(x)$. Then,

$$f_i(x) = \frac{1}{|I_i|} \sum_{j \in I_i} x_j = \frac{1}{|I_i|} \sum_{j \in I_i} f(x) = f(x).$$

Finally, the intensional derivative $\mathsf{D}\gamma$ is the function such that, for every $x \in \mathbb{R}^n$ and for i its partition index, satisfies $\mathsf{D}\gamma(x) = \mathsf{D}f_i(x)$, and hence

$$\mathsf{D}\gamma(x) = \mathsf{D}f_i(x) = \begin{pmatrix} a_1(x) & \cdots & a_n(x) \end{pmatrix} \qquad a_j(x) = \begin{cases} 1/|I_i| & \text{if } j \in I_i = I(x), \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the selected intensional derivative $\tilde{\mathsf{D}}_{\circledast} f$ is in fact the intensional derivative $\mathsf{D}\gamma$. □

Proposition 42. *The function $f(x) = \bigodot_{i=1}^{n} x_i$ is PAP, and $\tilde{\mathsf{D}}_{\circledast} f$ is an intensional derivative of f.*

Proof. Let $g(x) = \bigoplus_{i=1}^{n} x_i$. We have $f(x) = -g(-x)$, and hence f is PAP since it is the composition of PAP functions. Then, by the chain rule for PAP functions (Proposition 4), we have that

$$-\tilde{\mathsf{D}}_{\circledast} g(-x) \cdot \mathsf{D}(-x) \;=\; -\tilde{\mathsf{D}}_{\circledast} g(-x) \cdot (-\mathsf{D}x) \;=\; \tilde{\mathsf{D}}_{\circledast} g(-x)$$

is an intensional derivative of f. Let I_f and I_g be the active variable index functions of f and g, respectively, and let us note that

$$I_f(x) = \left\{ i \in [1..n] \mid \bigodot_{i=1}^{n} x_i = x_i \right\} = \left\{ i \in [1..n] \mid \bigoplus_{i=1}^{n} -x_i = -x_i \right\} = I_g(-x) \qquad \forall x \in \mathbb{R}^n.$$

Hence, we conclude that $\tilde{\mathsf{D}}_{\circledast} g(-x) = \tilde{\mathsf{D}}_{\circledast} f(x)$, and hence $\tilde{\mathsf{D}}_{\circledast} f$ is an intensional derivative of f. □

Proposition 43 (Intensional-admissibility of the AMM-PD-operator)**.** *Let G be a $\mathcal{P}_{\circledast}$-computation graph. Then, the pseudo-derivative operator $\tilde{\mathsf{D}}_{\circledast}$ is intensionally-admissible for G.*

Proof. By Definition 167, it suffices to show that $\tilde{\mathsf{D}}_{\circledast}$ is admissible for G and that, for every computation node $\mathring{v}$ of G, it holds that $\tilde{\mathsf{D}}_{\circledast}\check{v}$ is an intensional derivative of $\check{v}$.

Since every computation-node label of G belongs to $\mathcal{P}_{\circledast}$, Definition 228 implies immediately that $\tilde{\mathsf{D}}_{\circledast}$ is applicable to every computation-node label of G. Hence $\tilde{\mathsf{D}}_{\circledast}$ is admissible for G.

Let $\mathring{v}$ be an arbitrary computation node of G. By the definition of $\mathcal{P}_{\circledast}$, its label $\check{v}$ is one of the following:

- an analytic Euclidean function;
- a point-wise maximum function;
- a point-wise minimum function.

Suppose first that $\check{v}$ is analytic. Then $\check{v}$ is PAP, using the trivial PAP representation consisting of a single analytic piece, and its ordinary derivative $\mathsf{D}\check{v}$ is an intensional derivative of $\check{v}$. By Definition 228,

$$\tilde{\mathsf{D}}_{\circledast}\check{v} = \mathsf{D}\check{v}.$$

Therefore, $\tilde{\mathsf{D}}_{\circledast}\check{v}$ is an intensional derivative of $\check{v}$. This case includes zero-arity computation-node labels: by the convention for zero-arity functions, such a label is an analytic constant function

$$\check{v} : \mathbb{R}^0 \to \mathbb{R},$$

and

$$\tilde{\mathsf{D}}_{\circledast}\check{v} = \mathsf{D}\check{v} = 0_{1,0}.$$

Suppose next that $\check{v}$ is a point-wise maximum function. By Proposition 41,

$$\tilde{\mathsf{D}}_{\circledast}\check{v}$$

is an intensional derivative of $\check{v}$.

Finally, suppose that $\check{v}$ is a point-wise minimum function. By Proposition 42,

$$\tilde{\mathsf{D}}_{\circledast}\check{v}$$

is an intensional derivative of $\check{v}$.

Hence, for every computation-node label $\check{v}$ of G, the function $\tilde{\mathsf{D}}_{\circledast}\check{v}$ is an intensional derivative of $\check{v}$. Therefore, $\tilde{\mathsf{D}}_{\circledast}$ is intensionally-admissible for G. □

M.4 Analytic-MinMax Computation Graphs

Definition 229 (MinMax-algebraic computation-graph extensions)**.** Let $m \in \mathbb{N}^{>0}$, let $Q_1, \ldots, Q_m$ be computation graphs with common Euclidean input space Λ, and let

$$Q^{\sqcup} := Q_1 \sqcup_{\text{in}} \cdots \sqcup_{\text{in}} Q_m$$

be their common-input union as per Definition 151. Let V_{src} denote the set of scalar output nodes inherited from $Q_1, \ldots, Q_m$, called the *source-output nodes*.

A computation graph G is a *MinMax-algebraic extension* of $(Q_1, \ldots, Q_m)$ if it is obtained from $Q^{\sqcup}$ by adjoining fresh scalar computation nodes $W = \{\mathring{w}_1, \ldots, \mathring{w}_N\}$ while satisfying the following:

- for every $\ell \in [1..N]$:
  - node $\mathring{w}_\ell$ has label $\check{w}_\ell \in \mathcal{P}_{\circledast}$;
  - the predecessor node-vector $\mathring{v}_\ell$ of $\mathring{w}_\ell$ satisfies

$$\text{entries}(\mathring{v}_\ell) \subseteq V_{\text{src}} \cup \{\mathring{w}_1, \ldots, \mathring{w}_{\ell-1}\};$$

- for some $d_{\text{out}} \in \mathbb{N}^{>0}$, the output node-vector of G is $\mathring{v}^{\text{out}} \in (V_{\text{src}} \cup W)^{d_{\text{out}}}$.

We call $V_{\text{new}} := W$ the *new nodes*, and a node $\mathring{w} \in W$ is a *minimum node* or *maximum node* if its label is a Min or Max reduction, respectively.

We write $\mathcal{E}_{\circledast}(Q_1, \ldots, Q_m)$ for the class of MinMax-algebraic extensions of $(Q_1, \ldots, Q_m)$.
End of Definition 229.

Definition 230 (MinMax-algebraic virtual computation-graph extensions)**.** Let $m \in \mathbb{N}^{>0}$, let $Q_1, \ldots, Q_m$ be computation graphs with common Euclidean input space Λ, and let

$$Q^{\sqcup} := Q_1 \sqcup_{\text{in}} \cdots \sqcup_{\text{in}} Q_m$$

be their common-input union as per Definition 151. Let V_{src} denote the set of scalar output nodes inherited from $Q_1, \ldots, Q_m$, called the *source-output nodes*.

A virtual-computation graph G is a *MinMax-algebraic virtual extension* of $(Q_1, \ldots, Q_m)$ if it is obtained from $Q^{\sqcup}$ by adjoining fresh virtual computation nodes $W = \{\mathring{w}_1, \ldots, \mathring{w}_N\}$ in topological order while satisfying the following:

- for every $\ell \in [1..N]$:
  - node $\mathring{w}_\ell$ has label $\check{w}_\ell \in \mathcal{P}_{\circledast}$;
  - the predecessor node-vector $\mathring{v}_\ell$ of $\mathring{w}_\ell$ satisfies
$$\text{entries}(\mathring{v}_\ell) \subseteq V_{\text{src}} \cup \{\mathring{w}_1, \ldots, \mathring{w}_{\ell-1}\};$$
- for some $d_{\text{out}} \in \mathbb{N}^{>0}$, the output node-vector of G is $\mathring{v}^{\text{out}} \in \big(V_{\text{src}} \cup W\big)^{d_{\text{out}}}$.

We call $V^{\text{v}}_{\text{new}} := W$ the *virtual new nodes*, and a node $\mathring{w} \in W$ is a *minimum node* or *maximum node* if its label is a Min or Max reduction, respectively.

We write $\mathcal{E}^{\text{v}}_{\circledast}[Q_1, \ldots, Q_m]$ for the class of MinMax-algebraic virtual extensions of $(Q_1, \ldots, Q_m)$.
End of Definition 230.

Proposition 44 (Materialisation of MinMax-algebraic virtual extensions)**.** *Let $Q = (Q_1, \ldots, Q_m)$ be computation graphs with the same input arity, let $G \in \mathcal{E}^{\text{v}}_{\circledast}[Q]$, and let μ be a materialisation map for G, and let $G^\mu := G^\mu$. Then $G^\mu \in \mathcal{E}_{\circledast}(Q)$ and $\hat{G}^\mu = \hat{G}$.* End of Proposition 44.

Proof. Let $W = \{\mathring{w}_1, \ldots, \mathring{w}_N\}$ be the virtual new nodes of G. Since materialisation is the identity on the concrete nodes of G, it leaves the common-input union of $Q_1, \ldots, Q_m$ unchanged. By materialisation,

$$W^\mu := \{\mathring{w}^\mu_1, \ldots, \mathring{w}^\mu_N\}, \qquad \mathring{w}^\mu_\ell := \mu(\mathring{w}_\ell),$$

consists of fresh scalar computation nodes in the same topological order as $\mathring{w}_1, \ldots, \mathring{w}_N$, with

$$\check{w}^\mu_\ell = \check{w}_\ell \in \mathcal{P}_{\circledast} \qquad \forall \ell \in [1..N].$$

For every $\ell \in [1..N]$, let $\mathring{v}_\ell$ be the predecessor node-vector of $\mathring{w}_\ell$. By Definition 230,

$$\text{entries}(\mathring{v}_\ell) \subseteq V_{\text{src}} \cup \{\mathring{w}_1, \ldots, \mathring{w}_{\ell-1}\}.$$

Since materialisation is the identity on V_{src} and it maps predecessor node-vectors entry-wise, the predecessor node-vector of $\mathring{w}^\mu_\ell$ is $\bar{\mu}(\mathring{v}_\ell)$, and

$$\text{entries}(\bar{\mu}(\mathring{v}_\ell)) \subseteq V_{\text{src}} \cup \{\mathring{w}^\mu_1, \ldots, \mathring{w}^\mu_{\ell-1}\}.$$

Similarly, the materialised output node-vector belongs to $(V_{\text{src}} \cup W^\mu)^{d_{\text{out}}}$. Hence G^μ satisfies Definition 229, and therefore $G^\mu \in \mathcal{E}_{\circledast}(Q)$.

Finally, Proposition 21 gives $\hat{G}^\mu = \hat{G}$. □

Definition 231 (Analytic-MinMax computation graphs)**.** An *analytic-MinMax computation graph (AMM-graph)* is a $\mathcal{P}_{\circledast}$-computation graph. We denote by $\mathcal{G}_{\circledast}$ the class of all AMM-graphs. An *AMM-graph class* is a subset of $\mathcal{G}_{\circledast}$.

For every AMM-function q, we define

$$\mathcal{G}_{\circledast}[q] := \big\{G \in \mathcal{G}_{\circledast} \;\big|\; \hat{G} = q\big\};$$

and we call $\mathcal{G}_{\circledast}[q]$ the *AMM-graph family representing q* whenever $\mathcal{G}_{\circledast}[q] \neq \emptyset$. *End of Definition 231.*

Proposition 45 (MinMax extensions of AMM-graphs)**.** *Let $Q_1, \dots, Q_m$ be AMM-graphs. Then $\mathcal{E}_{\circledast}(Q_1, \dots, Q_m)$ is an AMM-graph class.* End of Proposition 45.

Proof. By Definition 223, every constant function and every finite minimum or maximum reduction belongs to $\mathcal{P}_{\circledast}$. Hence $\mathcal{P}_{\circledast} \subseteq \mathcal{P}_{\circledast}$. The computation-node labels inherited from $Q_1, \dots, Q_m$ belong to $\mathcal{P}_{\circledast}$ because these graphs are AMM-graphs, while every newly introduced node has a label in $\mathcal{P}_{\circledast}$. Therefore every computation-node label of every $G \in \mathcal{E}_{\circledast}(Q_1, \dots, Q_m)$ belongs to $\mathcal{P}_{\circledast}$, so G is an AMM-graph. □

Corollary 21. *Let $G \in \mathcal{E}_{\circledast}(Q_1, \dots, Q_m)$ be a MinMax-algebraic extension of AMM-graphs $Q_1, \dots, Q_m$, and let q be a Euclidean function. If G represents q, then $G \in \mathcal{G}_{\circledast}[q]$.*

Proof. Immediate consequence of Proposition 45. □

Proposition 46 (Characterisation of AMM-functions by AMM-graphs)**.** *For every function q,*

$$q \in \mathcal{F}_{\circledast} \qquad \Longleftrightarrow \qquad \mathcal{G}_{\circledast}[q] \neq \emptyset.$$

Proof. Suppose first that $q \in \mathcal{F}_{\circledast}$. By Definition 224, q is obtained by finitely many applications of function tupling and composition to functions in $\mathcal{P}_{\circledast}$. These operations can be represented by a finite computation graph whose computation-node labels belong to $\mathcal{P}_{\circledast}$. Hence there exists an AMM-graph G such that $\hat{G} = q$. Therefore, $G \in \mathcal{G}_{\circledast}[q]$, and hence $\mathcal{G}_{\circledast}[q] \neq \emptyset$.

Conversely, suppose that $G \in \mathcal{G}_{\circledast}[q]$. By Definition 231, every computation-node label of G belongs to $\mathcal{P}_{\circledast}$. Since G is finite, its value function is obtained by finitely many applications of function tupling and composition to functions in $\mathcal{P}_{\circledast}$. Hence, by Definition 224, we have $q = \hat{G} \in \mathcal{F}_{\circledast}$. □

M.5 Virtual Graph-Extension Notation for Analytic-Minax Graphs

Definition 232 (MinMax vector and matrix notation)**.** Let

$$G = \langle V, V_{\mathrm{v}}, E, \mathring{v}^{\mathrm{in}}, \mathring{v}^{\mathrm{c}}, \mathring{v}^{\mathrm{out}}, \lambda \rangle$$

be a virtual-computation graph.

We introduce notation for compactly specifying virtual-node extensions of G using node-vector and node-matrix presentations as per Definition 143.

A *presentation pattern* is, recursively, a scalar node, a node-vector presentation, a node-matrix presentation, or a finite tuple of presentation patterns. This is only syntactic notation for structured collections of scalar nodes and does not introduce an additional kind of graph object.

We write

$$G \xRightarrow[\mathrm{v}]{\mathcal{E}_1, \dots, \mathcal{E}_q} G'$$

when G' is the virtual-node extension of G obtained by successively applying the expressions

$$\mathcal{E}_1, \dots, \mathcal{E}_q$$

from left to right, according to the rules below. Thus, every expression may use nodes introduced by the preceding expressions in the same extension.

Every scalar node introduced by this notation is a fresh virtual computation node, and all virtual nodes introduced in a single extension are pair-wise distinct and distinct from the nodes already present in G.

The expressions are interpreted as follows.

- *Scalar copy.* For every node $\mathring{u}$ available in the current virtual-computation graph, an expression

  $$\mathring{w} := \mathring{u}$$

  denotes the virtual-node extension from Definition 140 obtained by adjoining $\mathring{w}$ with predecessor node-vector $(\mathring{u})$ and label $\mathrm{id}_{\mathbb{R}}$.

- *Vector and matrix copies.* If $\mathring{u}$ and $\mathring{w}$ are node-vector presentations of the same dimension, then
$$\mathring{w} := \mathring{u}$$
is interpreted entrywise by the scalar-copy rule. Likewise, if $\mathring{A}$ and $\mathring{B}$ are node-matrix presentations of the same dimensions, then
$$\mathring{B} := \mathring{A}$$
is interpreted entrywise by the scalar-copy rule.

- *Tuple copies.* If
$$\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_r)$$
is a presentation pattern, then an expression
$$\mathcal{Q} := \mathcal{P},$$
where $\mathcal{Q}$ has the same tuple, vector, and matrix structure as $\mathcal{P}$, is interpreted recursively by applying the scalar-, vector-, and matrix-copy rules to its components. In particular,
$$\mathring{M} := (\mathring{A}, \mathring{b})$$
introduces fresh virtual copies of all scalar entries of $\mathring{A}$ and $\mathring{b}$, arranged with the same matrix–vector structure.

- *Fresh vector and matrix presentations.* An expression consisting only of a node-vector or node-matrix symbol, such as
$$\mathring{z} \qquad \text{or} \qquad \mathring{A},$$
denotes a fresh virtual node-vector or node-matrix presentation, respectively, provided that the surrounding specification gives, for every scalar entry, its predecessor node-vector and scalar label. The corresponding scalar virtual nodes are adjoined successively according to Definition 140.

- *Minimum and maximum reductions.* Let
$$\mathring{u} = (\mathring{u}_1, \ldots, \mathring{u}_r)$$
be a node-vector presentation available in the current graph, with $r \geq 2$. Then
$$\mathring{w} := \mathsf{min}(\mathring{u})$$
denotes the virtual-node extension obtained by adjoining $\mathring{w}$ with predecessor node-vector $\mathring{u}$ and label min_r. Analogously,
$$\mathring{w} := \mathsf{max}(\mathring{u})$$
denotes the extension obtained by adjoining $\mathring{w}$ with predecessor node-vector $\mathring{u}$ and label max_r.

- *Component-wise maximum.* Let
$$\mathring{x}, \mathring{y} \in (V \cup V_{\mathrm{v}})^n$$
be node-vector presentations available in the current graph. Then
$$\mathring{z} := \mathring{x} \oplus \mathring{y}$$
denotes the successive introduction, for every $i \in [1..n]$, of a fresh virtual node $\mathring{z}_i$ with predecessor node-vector
$$((\mathring{x})_i, (\mathring{y})_i)$$
and label max_2.

- *MinMax matrix–vector product.* Let
$$\mathring{A} \in (V \cup V_{\mathrm{v}})^{m \times n}, \qquad \mathring{x} \in (V \cup V_{\mathrm{v}})^n$$
be node-matrix and node-vector presentations available in the current graph. Then
$$\mathring{z} := \mathring{A} \otimes \mathring{x}$$
denotes the following successive virtual-node extensions:

- for every $i \in [1..m]$ and $j \in [1..n]$, introduce a fresh virtual node $\mathring{w}_{i,j}$ with predecessor node-vector
$$((\mathring{A})_{i,j}, (\mathring{x})_j)$$
and label min_2;
- for every $i \in [1..m]$, introduce a fresh virtual node $\mathring{z}_i$ with predecessor node-vector
$$(\mathring{w}_{i,1}, \ldots, \mathring{w}_{i,n})$$
and label max_n.

- *MinMax matrix–matrix product.* Let
$$\mathring{A} \in (V \cup V_{\mathrm{v}})^{m \times n}, \qquad \mathring{B} \in (V \cup V_{\mathrm{v}})^{n \times p}$$
be node-matrix presentations available in the current graph. Then
$$\mathring{C} := \mathring{A} \otimes \mathring{B}$$
denotes the following successive virtual-node extensions:
  - for every $i \in [1..m]$, $j \in [1..p]$, and $r \in [1..n]$, introduce a fresh virtual node $\mathring{w}_{i,j,r}$ with predecessor node-vector
  $$((\mathring{A})_{i,r}, (\mathring{B})_{r,j})$$
  and label min_2;
  - for every $i \in [1..m]$ and $j \in [1..p]$, introduce a fresh virtual node $\mathring{C}_{i,j}$ with predecessor node-vector
  $$(\mathring{w}_{i,j,1}, \ldots, \mathring{w}_{i,j,n})$$
  and label max_n.

Whenever the right-hand side of an expression contains several MinMax operations, they are expanded recursively in the order determined by the expression, with every intermediate scalar result represented by a fresh virtual computation node.

In particular, several expressions may be combined in a single extension, as in
$$G \xrightarrow[\mathrm{v}]{\mathring{v}_{\mathsf{min}} := \mathsf{min}(\mathring{\mathcal{V}}),\ \mathring{v}_{\mathsf{max}} := \mathsf{max}(\mathring{\mathcal{V}}),\ \mathring{E},\ \mathring{e},\ \mathring{M}_{\mathrm{id}} := (\mathring{E}, \mathring{e})} G',$$
where component-wise predecessor node-vectors and labels for $\mathring{E}$ and $\mathring{e}$ may be specified separately.

End of Definition 232.

Fact 48 (Value functions of MinMax presentations)**.** *Assume the notation of Definition 232, and let*
$$G \xrightarrow[\mathrm{v}]{\mathcal{E}_1, \ldots, \mathcal{E}_q} G'$$
be an extension specified by that notation. Then every scalar, node-vector, and node-matrix presentation introduced by an expression $\mathcal{E}_j$ has the value function obtained by evaluating its right-hand side point-wise.

In particular, for every $u \in \mathbb{R}^{d_{\mathrm{in}}}$, the following hold whenever the corresponding expressions occur in the extension:
$$\begin{aligned}
\mathring{z} := \mathring{x} \quad &\Longrightarrow \quad \hat{z}(u) = \hat{x}(u), \\
\mathring{z} := \mathsf{min}(\mathring{x}) \quad &\Longrightarrow \quad \hat{z}(u) = \mathsf{min}_n\big(\hat{x}(u)\big), \\
\mathring{z} := \mathsf{max}(\mathring{x}) \quad &\Longrightarrow \quad \hat{z}(u) = \mathsf{max}_n\big(\hat{x}(u)\big), \\
\mathring{z} := \mathring{x} \oplus \mathring{y} \quad &\Longrightarrow \quad \hat{z}(u) = \hat{x}(u) \oplus \hat{y}(u), \\
\mathring{z} := \mathring{A} \otimes \mathring{x} \quad &\Longrightarrow \quad \hat{z}(u) = \hat{A}(u) \otimes \hat{x}(u), \\
\mathring{C} := \mathring{A} \otimes \mathring{B} \quad &\Longrightarrow \quad \hat{C}(u) = \hat{A}(u) \otimes \hat{B}(u).
\end{aligned}$$
Here, in the second and third lines, $\mathring{x}$ has dimension n.

The first implication applies component-wise to node-vector and node-matrix copies. More generally, the value of every structured presentation introduced by the notation is obtained component-wise from the values of the presentations occurring on its right-hand side.

If several expressions occur in the same extension, the implications above apply successively in their order of occurrence. In particular, later expressions may use presentations introduced by earlier expressions, and all corresponding value identities hold simultaneously in G'. End of Fact 48.

Remark 26 (MinMax notation and materialisation). The notation of Definition 232 is shorthand for successive virtual-node extensions in the sense of Definition 140. In particular, an expression

$$G \xRightarrow[\mathrm{v}]{\mathcal{E}_1,\ldots,\mathcal{E}_q} G'$$

specifies the extended virtual-computation graph G' together with all scalar, node-vector, and node-matrix presentations introduced by the expressions $\mathcal{E}_1, \ldots, \mathcal{E}_q$.

Moreover, suppose that, for computation graphs $Q_1, \ldots, Q_m$,

$$G \in \mathcal{E}^{\mathrm{v}}_{\circledast}[Q_1, \ldots, Q_m].$$

Since every scalar virtual computation node introduced by the notation has a label admitted by Definition 230, and its predecessors belong to the source nodes or to previously introduced extension nodes, we also have

$$G' \in \mathcal{E}^{\mathrm{v}}_{\circledast}[Q_1, \ldots, Q_m].$$

Consequently, by Proposition 44, for every materialisation map μ of G',

$$G'^{\mu} \in \mathcal{E}_{\circledast}(Q_1, \ldots, Q_m)\,, \qquad \hat{G'^{\mu}} = \hat{G'}.$$

In particular, materialisation preserves component-wise the value functions of every node-vector and node-matrix presentation introduced by the notation.

Finally, if $Q_1, \ldots, Q_m$ are AMM-graphs, then Proposition 45 implies that G'^{μ} is an AMM-graph; consequently, by Proposition 46, its value function is an AMM-function. *End of Remark 26.*

M.6 Properties of the Analytic-MinMax Pseudo-Derivative Operator

Definition 233 (AMM computation-graph family)**.** Let $\mathcal{Q} \subseteq \mathcal{F}_{\circledast}$. We define the *AMM computation-graph family for* $\mathcal{Q}$ by

$$\mathcal{G}_{\circledast}[\mathcal{Q}] := \big(\mathcal{G}_{\circledast}[q]\,\big)_{q\in\mathcal{Q}}.$$

End of Definition 233.

Fact 49. *For every $\mathcal{Q} \subseteq \mathcal{F}_{\circledast}$, the family $\mathcal{G}_{\circledast}[\mathcal{Q}]$ is a $\mathcal{P}_{\circledast}$-computation-graph family for $\mathcal{Q}$.* End of Fact 49.

Proof. Let $q \in \mathcal{Q}$. Proposition 46 ensures $\mathcal{G}_{\circledast}[q] \neq \emptyset$. Furthermore, by definition, every graph $G \in \mathcal{G}_{\circledast}[q]$ is a $\mathcal{P}_{\circledast}$-computation graph representing q. Hence $\mathcal{G}_{\circledast}[\mathcal{Q}]$ is a $\mathcal{P}_{\circledast}$-computation-graph family for $\mathcal{Q}$. □

Theorem 45 (SID-operators for AMM-functions)**.** *Let $\mathcal{Q} \subseteq \mathcal{F}_{\circledast}$, let $\tilde{\mathsf{D}}_{\circledast}$ be the analytic-MinMax pseudo-derivative operator,*[12] *and let Γ be a graph selector for $\mathcal{Q}$ over $\mathcal{G}_{\circledast}[\mathcal{Q}]$. Then the graph-induced pseudo-derivative operator $\tilde{\mathsf{D}}^{\Gamma}_{\circledast}$ is a selected intensional-derivative operator on $\mathcal{Q}$.*

Proof. Since Γ is a graph selector for $\mathcal{Q}$ over $\mathcal{G}_{\circledast}[\mathcal{Q}]$, we have

$$\Gamma(q) \in \mathcal{G}_{\circledast}[q] \subseteq \mathcal{G}_{\circledast} \qquad \forall q \in \mathcal{Q}.$$

Hence, by Definition 231, every computation-node label of every graph in $\mathrm{Im}(\Gamma)$ belongs to $\mathcal{P}_{\circledast}$. Proposition 43 therefore implies that $\tilde{\mathsf{D}}_{\circledast}$ is intensionally-admissible for every graph in $\mathrm{Im}(\Gamma)$, and hence for $\mathrm{Im}(\Gamma)$. The claim follows from Proposition 25. □

[12] The operator is introduced in Definition 228.

Lemma 31 (Main convexity lemma)**.** *Let $\tilde{\mathsf{D}}_{\circledast}$ be the pseudo-derivative operator of Definition 228, and let*

$$G \in \mathcal{E}_{\circledast}(Q_1, \dots, Q_m)$$

be a MinMax-algebraic extension with source-output node set V_{src} and new-node set V_{new}. For every $i \in [1..m]$, let d_i be the output arity of Q_i. and let $Q_{i,j} := Q_i|_j$ be the output-restriction of Q_i to coordinate $j \in [1..d_i]$.

Then, for every $\mathring{v} \in V_{\mathrm{src}} \cup V_{\mathrm{new}}$ and every $\lambda \in \Lambda$, there exist non-negative coefficients

$$\alpha^{\mathring{v}}_{i,j}(\lambda) \in \mathbb{R}^{\geq 0} \qquad \textit{for } i \in [m], j \in [d_i],$$

such that

$$\sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha^{\mathring{v}}_{i,j}(\lambda) = 1 \qquad \textit{and} \qquad \tilde{\mathsf{D}}_{\circledast} \mathring{v}(\lambda) = \sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha^{\mathring{v}}_{i,j}(\lambda) \cdot \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda)\,. \tag{95}$$

Consequently,

$$\left\| \tilde{\mathsf{D}}_{\circledast} G(\lambda) \right\|_{\mathrm{row}} \leq \max_{i \in [1..m]} \left\| \tilde{\mathsf{D}}_{\circledast} Q_i(\lambda) \right\|_{\mathrm{row}}. \tag{96}$$

Proof. We first consider the source-output nodes. Let $\mathring{v} \in V_{\mathrm{src}}$ be the j-th scalar output node inherited from Q_i, and define

$$\alpha^{\mathring{v}}_{i',j'}(\lambda) := \begin{cases} 1 & \text{if } (i',j') = (i,j), \\ 0 & \text{if } (i',j') \neq (i,j). \end{cases}$$

Then

$$\sum_{i'=1}^{m} \sum_{j'=1}^{d_{i'}} \alpha^{\mathring{v}}_{i',j'}(\lambda) = 1,$$

and, since $\mathring{v}$ is the corresponding scalar output node of Q_i,

$$\tilde{\mathsf{D}}_{\circledast} \mathring{v}(\lambda) = \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda).$$

Hence (95) holds for every source-output node.

We now proceed by induction over the topological ordering over the new nodes

$$\mathring{w}_1, \dots, \mathring{w}_N \in V_{\mathrm{new}}.$$

Let $\ell \in [1..N]$, and suppose that the claim holds for every $\mathring{w}_r$ with $r \in [1..\ell-1]$. Let $\mathring{v} := \mathring{w}_\ell$ have arity k and ordered predecessor tuple

$$(\mathring{v}_1, \dots, \mathring{v}_k).$$

By Definition 229, every predecessor satisfies

$$\mathring{v}_r \in V_{\mathrm{src}} \cup \{\mathring{w}_1, \dots, \mathring{w}_{\ell-1}\}, \qquad r \in [1..k].$$

Hence, by the source-output case or the induction hypothesis, for every $r \in [1..k]$ there exist coefficients

$$\alpha^{\mathring{v}_r}_{i,j}(\lambda) \in \mathbb{R}^{\geq 0}$$

satisfying

$$\sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha^{\mathring{v}_r}_{i,j}(\lambda) = 1 \tag{97}$$

and

$$\tilde{\mathsf{D}}_{\circledast} \mathring{v}_r(\lambda) = \sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha^{\mathring{v}_r}_{i,j}(\lambda) \cdot \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda). \tag{98}$$

Since $\mathring{v}$ is either a minimum node or a maximum node, Definition 228 and Propositions 38 and 39 imply that the local pseudo-derivative of its label is a convex combination of the coordinate derivatives. Hence there exist

$$\beta_1(\lambda), \dots, \beta_k(\lambda) \in \mathbb{R}^{\geq 0}$$

such that

$$\sum_{r=1}^{k} \beta_r(\lambda) = 1. \tag{99}$$

Therefore, by the definition of graph-induced pseudo-derivatives in Definition 163,

$$\tilde{\mathsf{D}}_{\circledast} \mathring{v}(\lambda) = \sum_{r=1}^{k} \beta_r(\lambda) \cdot \tilde{\mathsf{D}}_{\circledast} \mathring{v}_r(\lambda). \tag{100}$$

Substituting (98) into (100) gives

$$\begin{aligned}\tilde{\mathsf{D}}_{\circledast} \mathring{v}(\lambda) &= \sum_{r=1}^{k} \beta_r(\lambda) \cdot \sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha_{i,j}^{\mathring{v}_r}(\lambda) \cdot \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{d_i} \left(\sum_{r=1}^{k} \beta_r(\lambda) \cdot \alpha_{i,j}^{\mathring{v}_r}(\lambda) \right) \cdot \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda).\end{aligned}$$

Define

$$\alpha_{i,j}^{\mathring{v}}(\lambda) := \sum_{r=1}^{k} \beta_r(\lambda) \cdot \alpha_{i,j}^{\mathring{v}_r}(\lambda) \qquad \forall i \in [1..m],\ \forall j \in [1..d_i].$$

These coefficients are non-negative, and by (97) and (99),

$$\sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha_{i,j}^{\mathring{v}}(\lambda) \;=\; \sum_{r=1}^{k} \beta_r(\lambda) \cdot \sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha_{i,j}^{\mathring{v}_r}(\lambda) \;=\; \sum_{r=1}^{k} \beta_r(\lambda) \;=\; 1.$$

Thus Eq. (95) holds for $\mathring{w}_\ell$. Hence, by induction over $\ell \in [1..N]$, it holds for every node in V_{new}. Together with the source-output case proved above, it therefore holds for every node in $V_{\mathrm{src}} \cup V_{\mathrm{new}}$.

Finally, let $\mathring{v}$ be a scalar output node of G. By Definition 229, we have $\mathring{v} \in V_{\mathrm{src}} \cup V_{\mathrm{new}}$. Hence, by Eq. (95) and the triangle inequality, we obtain

$$\left\| \tilde{\mathsf{D}}_{\circledast} \mathring{v}(\lambda) \right\| \;\le\; \sum_{i=1}^{m} \sum_{j=1}^{d_i} \alpha_{i,j}^{\mathring{v}}(\lambda) \left\| \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda) \right\| \;\le\; \max_{\substack{i \in [1..m] \\ j \in [1..d_i]}} \left\| \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda) \right\|.$$

Taking the maximum over the output nodes of G gives

$$\left\| \tilde{\mathsf{D}}_{\circledast} G(\lambda) \right\|_{\mathrm{row}} \;\le\; \max_{\substack{i \in [1..m] \\ j \in [1..d_i]}} \left\| \tilde{\mathsf{D}}_{\circledast} Q_{i,j}(\lambda) \right\| \;=\; \max_{i \in [1..m]} \left\| \tilde{\mathsf{D}}_{\circledast} Q_i(\lambda) \right\|_{\mathrm{row}},$$

proving (96). □

M.7 MinMax Recurrence Computation Graphs

Definition 234 (MinMax-algebraic computation graphs for a MinMax recurrence)**.** Let $x_0 \in \mathbb{R}^n$, and let $T \in \mathbb{N}^{>0}$. For every $t \in [1..T]$, let us consider AMM-functions $R_t, s_t \in \mathcal{F}_{\circledast}$ with

$$R_t : \Lambda \to \mathbb{R}^{n \times n}, \qquad s_t : \Lambda \to \mathbb{R}^n,$$

and corresponding AMM-graphs $\mathring{R}_t \in \mathcal{G}_{\circledast}[R_t]$ and $\mathring{s}_t \in \mathcal{G}_{\circledast}[s_t]$.

Let $x_0 : \Lambda \to \mathbb{R}^n$ be the constant function $x_0(\lambda) := x_0$ with canonical graph $\mathring{x}_0 := G_{\mathrm{c}}^{\mathrm{p}}[x_0, \Lambda]$, and let $(x_t)_{t=1}^{T}$ be the sequence given recursively by

$$x_t(\lambda) := \big(R_t(\lambda) \otimes x_{t-1}(\lambda) \big) \oplus s_t(\lambda) \qquad \forall t \in [1..T].$$

The corresponding *MinMax-algebraic recurrence computation-graph family* is

$$\mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big) := \big\{ G \in \mathcal{E}_{\circledast}\big[\mathring{x}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big] \;\big|\; \hat{G} = x_T^u \big\}.$$

End of Definition 234.

Remark 27. The qualification *MinMax-algebraic* of the recurrence graphs in Definition 234 is meant to emphasise that they are obtained through MinMax-algebraic extensions. In particular, we do not introduce a separate notion of *MinMax-algebraic computation graph* or *MinMax-algebraic computation-graph class*.

Instead, we show below in Proposition 47 that, whenever the source graphs are Analytic-MinMax, the resulting MinMax-algebraic recurrence graphs are themselves Analytic-MinMax computation graphs. In particular, this applies to the recurrence graphs arising in MinMax RNCs, where the functions used outside the MinMax recurrence are analytic. *End of Remark 27.*

Proposition 47 (Analytic-MinMax closure of MinMax-algebraic recurrence graphs)**.** *Assume the context of Definition 234. Then*

$$\mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big) \subseteq \mathcal{G}_{\circledast}[x_T]\,.$$

Proof. Let

$$G \in \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big).$$

By Definition 234, we have

$$G \in \mathcal{E}_{\circledast}\big[\mathring{x}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big] \qquad \text{and} \qquad \hat{G} = x_T.$$

For every $t \in [1..T]$, we have $\mathring{R}_t \in \mathcal{G}_{\circledast}[R_t]$ and $\mathring{s}_t \in \mathcal{G}_{\circledast}[s_t]$, and hence $\mathring{R}_t$ and $\mathring{s}_t$ are AMM-graphs. Furthermore, $\mathring{x}_0$ is a constant computation graph, whose computation-node labels are constant and therefore analytic functions; hence $\mathring{x}_0$ is also an AMM-graph.

Therefore, by Proposition 45, G is an AMM-graph. Since $\hat{G} = x_T$, we conclude that $G \in \mathcal{G}_{\circledast}[x_T]$. Since G was arbitrary, the claimed inclusion follows. □

Theorem 46 (Topology-independent derivative bound for MinMax recurrence evaluation)**.** *Assume the context of Definition 234, and let*

$$G \in \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big).$$

Then, for every $\lambda \in \Lambda$,

$$\big\|\tilde{\mathsf{D}}_{\circledast} G(\lambda)\big\|_{\mathrm{row}} \leq \max_{t\in[1..T]} \max\Big\{\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t(\lambda)\big\|_{\mathrm{row}}, \big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t(\lambda)\big\|_{\mathrm{row}}\Big\}\,.$$

Proof. By Definition 234,

$$G \in \mathcal{E}_{\circledast}\big[\mathring{x}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\big].$$

Hence, Lemma 31 gives

$$\big\|\tilde{\mathsf{D}}_{\circledast} G(\lambda)\big\|_{\mathrm{row}} \leq \max\left\{\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{x}_0(\lambda)\big\|_{\mathrm{row}}, \max_{t\in[1..T]}\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t(\lambda)\big\|_{\mathrm{row}}, \max_{t\in[1..T]}\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t(\lambda)\big\|_{\mathrm{row}}\right\}.$$

Since $\mathring{x}_0$ represents a constant function of λ, Lemma 6 gives

$$\tilde{\mathsf{D}}_{\circledast}\mathring{x}_0(\lambda) = 0.$$

Therefore,

$$\begin{aligned}\big\|\tilde{\mathsf{D}}_{\circledast} G(\lambda)\big\|_{\mathrm{row}} &\leq \max\left\{\max_{t\in[1..T]}\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t(\lambda)\big\|_{\mathrm{row}}, \max_{t\in[1..T]}\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t(\lambda)\big\|_{\mathrm{row}}\right\} \\ &= \max_{t\in[1..T]}\max\Big\{\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t(\lambda)\big\|_{\mathrm{row}}, \big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t(\lambda)\big\|_{\mathrm{row}}\Big\}\,,\end{aligned}$$

proving the claim. □

M.8 MinMaxRNC State Computation Graphs

Definition 235 (Identity parameter computation graph)**.** Let $d_\Theta \in \mathbb{N}$, and let

$$\mathring{\theta} = (\mathring{\theta}_1, \dots, \mathring{\theta}_{d_\Theta})$$

be a node-vector.

The *identity parameter computation graph* $\mathring{I}[\mathring{\theta}]$ is the computation graph having no computation nodes and satisfying

$$\mathring{v}^{\mathrm{in}} = \mathring{v}^{\mathrm{out}} = \mathring{\theta}.$$

End of Definition 235.

Fact 50. *In the context of Definition 235, we have that $\mathring{I}[\mathring{\theta}]$ represents the identity function on $\mathbb{R}^{d_\Theta}$.*

End of Fact 50.

Definition 236 (MinMax-algebraic RNC state computation-graph class)**.** We define MinMax-algebraic computation graphs for the states of MinMax RNCs.

Context. Let us consider RNC dynamics $D \in \mathrm{RNC}_{\mathrm{dyn}}\big(\mathcal{U}^{\mathrm{p}}_{\mathrm{AN}}[\mathrm{MinMax}]\big)$ having state space X, input space U, parameter space $\Theta \subseteq \mathbb{R}^{d_\Theta}$, and cascade decomposition

$$D = D_1 \ltimes_{g_1} \cdots \ltimes_{g_{L-1}} D_L, \qquad D_\ell = D_{\ell,1} \parallel \cdots \parallel D_{\ell,n_\ell}.$$

Let $u \in U^\omega$, $x_0 \in X$, and $T \in \mathbb{T}$.

Auxiliary. Let

$$\mathring{\theta} = (\mathring{\theta}_1, \dots, \mathring{\theta}_{d_\Theta})$$

be a node-vector. All computation graphs introduced below have $\mathring{\theta}$ as their input node-vector. In particular, all source graphs combined by common-input unions or smooth graph extensions have this same input node-vector. The smooth extensions below use no additional parameter input, since the parameter value is supplied explicitly by $\mathring{I}[\mathring{\theta}]$.

For every $\ell \in [1..L]$, let X_ℓ and U_ℓ denote the state and input spaces of D_ℓ, with $U_1 = U$, and, for every $k \in [1..n_\ell]$, let $X_{\ell,k}$ be the state space of $D_{\ell,k}$. We identify

$$X_\ell = X_{\ell,1} \times \cdots \times X_{\ell,n_\ell}, \qquad X = X_1 \times \cdots \times X_L.$$

For every $\ell \in [1..L]$ and $k \in [1..n_\ell]$,

- let $\rho_{\ell,k}$ be the function

$$\rho_{\ell,k} : U_\ell \times \Theta \to \mathbb{R}^{d_R^{\ell,k} + d_s^{\ell,k}}, \qquad \rho_{\ell,k}\big(u, \theta\big) := \big(R_{\ell,k}\big(u; \theta\big), s_{\ell,k}\big(u; \theta\big)\big),$$

  where $d_R^{\ell,k}$ and $d_s^{\ell,k}$ are the output arities of $R_{\ell,k}$ and $s_{\ell,k}$, respectively;

- let $J_R^{\ell,k}$ and $J_s^{\ell,k}$ be the sets of indices

$$J_R^{\ell,k} := (1, \dots, d_R^{\ell,k}), \qquad J_s^{\ell,k} := (d_R^{\ell,k} + 1, \dots, d_R^{\ell,k} + d_s^{\ell,k}).$$

Let us write

$$x_0 = (x_0^1, \dots, x_0^L), \qquad x_0^\ell = (x_0^{\ell,1}, \dots, x_0^{\ell,n_\ell}).$$

Definition. For $T = 0$, we define

$$\mathcal{G}_{\circledast}(D; 0, u, x_0) := \big\{G^{\mathrm{p}}_{\mathrm{c}}[x_0, \mathring{\theta}]\big\}.$$

For $T \in \mathbb{N}^{>0}$, the *MinMax-algebraic RNC state computation-graph class* $\mathcal{G}_{\circledast}(D; T, u, x_0)$ is the class of computation graphs

$$G = \mathring{X}_T^1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_T^L$$

for which there exist computation graphs

$$\mathring{U}_t^\ell, \quad \mathring{R}_t^{\ell,k}, \quad \mathring{s}_t^{\ell,k}, \quad \mathring{H}_t^{\ell,k}, \quad \mathring{X}_t^{\ell,k}, \quad \mathring{X}_t^\ell,$$

called *constituent graphs* of G, such that:

$$\begin{aligned}
\mathring{X}_0^{\ell} &= G_{\mathrm{c}}^{\mathrm{p}}[x_0^{\ell}, \mathring{\theta}] && \forall \ell \in [1..L], \\
\mathring{U}_t^{1} &= G_{\mathrm{c}}^{\mathrm{p}}[u_t, \mathring{\theta}] && \forall t \in [1..T], \\
\mathring{H}_t^{\ell,k} &= (\mathring{U}_t^{\ell}, \mathring{I}[\mathring{\theta}]) \triangleright \rho_{\ell,k} && \forall \ell \in [1..L],\ \forall k \in [1..n_\ell],\ \forall t \in [1..T], \\
\mathring{R}_t^{\ell,k} &= \big(\mathring{H}_t^{\ell,k}\big)_{J_R^{\ell,k}}, && \forall \ell \in [1..L],\ \forall k \in [1..n_\ell],\ \forall t \in [1..T], \\
\mathring{s}_t^{\ell,k} &= \big(\mathring{H}_t^{\ell,k}\big)_{J_s^{\ell,k}} && \forall \ell \in [1..L],\ \forall k \in [1..n_\ell],\ \forall t \in [1..T], \\
\mathring{X}_t^{\ell,k} &\in \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0^{\ell,k}; \mathring{R}_{[1..t]}^{\ell,k}, \mathring{s}_{[1..t]}^{\ell,k},\big) && \forall \ell \in [1..L],\ \forall k \in [1..n_\ell],\ \forall t \in [1..T], \\
\mathring{X}_t^{\ell} &= \mathring{X}_t^{\ell,1} \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_t^{\ell,n_\ell} && \forall \ell \in [1..L],\ \forall t \in [1..T] \\
\mathring{U}_t^{\ell+1} &= \big(\mathring{X}_{t-1}^{\ell}, \mathring{U}_t^{\ell}, \mathring{X}_t^{\ell}, \mathring{I}[\mathring{\theta}]\big) \triangleright g_\ell && \forall \ell \in [1..L-1],\ \forall t \in [1..T].
\end{aligned}$$

Finally, we define the following aggregate classes:

$$\begin{aligned}
\mathcal{G}_{\circledast}(D;T,u) &:= \bigcup_{x_0 \in X} \mathcal{G}_{\circledast}(D;T,u,x_0)\,, \\
\mathcal{G}_{\circledast}(D;T) &:= \bigcup_{u \in U^{\omega}} \mathcal{G}_{\circledast}(D;T,u)\,, \\
\mathcal{G}_{\circledast}(D) &:= \bigcup_{T \in \mathbb{T}} \mathcal{G}_{\circledast}(D;T)\,.
\end{aligned}$$

End of Definition 236.

Remark 28 (Choices of constituent state graphs). A graph

$$G \in \mathcal{G}_{\circledast}(D;T,u,x_0)$$

need not uniquely determine the constituent computation graphs satisfying Definition 236. In particular, for each unit and each horizon, the definition permits an arbitrary recurrence graph from the corresponding class $\mathcal{G}_{\mathrm{rec},\circledast}$. Consequently, different admissible choices of recurrence graphs may give rise to the same terminal computation graph G.

This non-uniqueness is a feature of the general MinMax-algebraic state-graph class and should be distinguished from the specific sequential and parallel-scan constructions introduced below. As discussed in Remarks 30 and 32, once their source graphs are fixed, these constructions determine their recurrence computation graphs, subject in the parallel-scan case to the fixed choice of materialisation.

End of Remark 28.

Proposition 48 (Closure of MinMax RNC state graphs under analytic-MinMax computation)**.** *Assume the context of Definition 236, and let*

$$G \in \mathcal{G}_{\circledast}(D;T,u,x_0)\,.$$

Then G is an AMM-graph.

Furthermore, if $T \in \mathbb{N}^{>0}$, then, for every choice of constituent computation graphs

$$\mathring{U}_t^{\ell}, \qquad \mathring{R}_t^{\ell,k}, \qquad \mathring{s}_t^{\ell,k}, \qquad \mathring{H}_t^{\ell,k}, \qquad \mathring{X}_t^{\ell,k}, \qquad \mathring{X}_t^{\ell}$$

satisfying the defining conditions of Definition 236 for G, every such constituent graph is an AMM-graph.

Proof. Suppose first that $T = 0$. By Definition 236,

$$G = G_{\mathrm{c}}^{\mathrm{p}}[x_0, \mathring{\theta}].$$

Its computation-node labels are constant and therefore analytic, so G is an AMM-graph.

Suppose now that $T \in \mathbb{N}^{>0}$, and fix any choice of constituent computation graphs satisfying the defining conditions of Definition 236 for G. We proceed inductively over the layers.

The identity graph

$$\mathring{I}[\mathring{\theta}]$$

has no computation nodes and is therefore an AMM-graph.

For every $t \in [1..T]$, Definition 236 gives

$$\mathring{U}_t^1 = G_{\mathrm{c}}^{\mathrm{p}}[u_t, \mathring{\theta}],$$

and hence $\mathring{U}_t^1$ is an AMM-graph.

Let $\ell \in [1..L]$ and suppose that $\mathring{U}_t^\ell$ is an AMM-graph for every $t \in [1..T]$. For every $k \in [1..n_\ell]$, the function

$$\rho_{\ell,k} : U_\ell \times \Theta \to \mathbb{R}^{d_R^{\ell,k}+d_s^{\ell,k}}$$

is analytic by the definition of MinMax RNCs. Therefore, by Fact 47, for every $t \in [1..T]$,

$$\mathring{H}_t^{\ell,k} = (\mathring{U}_t^\ell, \mathring{I}[\mathring{\theta}]) \triangleright \rho_{\ell,k}$$

is an AMM-graph.

Output restriction introduces no new computation nodes. Hence

$$\mathring{R}_t^{\ell,k} = \big(\mathring{H}_t^{\ell,k}\big)_{J_R^{\ell,k}}, \qquad \mathring{s}_t^{\ell,k} = \big(\mathring{H}_t^{\ell,k}\big)_{J_s^{\ell,k}}$$

are AMM-graphs for every $t \in [1..T]$.

Furthermore, for every $t \in [1..T]$,

$$\mathring{X}_t^{\ell,k} \in \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0^{\ell,k}; \mathring{R}_{[1..t]}^{\ell,k}, \mathring{s}_{[1..t]}^{\ell,k}\big).$$

Hence, by Proposition 47, every $\mathring{X}_t^{\ell,k}$ is an AMM-graph.

Since common-input union introduces no new computation-node labels, every

$$\mathring{X}_t^\ell = \mathring{X}_t^{\ell,1} \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_t^{\ell,n_\ell}$$

is an AMM-graph.

If $\ell < L$, then

$$\mathring{X}_0^\ell = G_{\mathrm{c}}^{\mathrm{p}}[x_0^\ell, \mathring{\theta}]$$

is an AMM-graph. Furthermore, g_ℓ is analytic by the definition of the RNC cascade composition. Thus, by Fact 47, for every $t \in [1..T]$,

$$\mathring{U}_t^{\ell+1} = \big(\mathring{X}_{t-1}^\ell, \mathring{U}_t^\ell, \mathring{X}_t^\ell, \mathring{I}[\mathring{\theta}]\big) \triangleright g_\ell$$

is an AMM-graph. This closes the induction over layers.

Finally, by Definition 236,

$$G = \mathring{X}_T^1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_T^L.$$

Since each $\mathring{X}_T^\ell$ is an AMM-graph and common-input union introduces no new computation-node labels, G is an AMM-graph.

The argument simultaneously establishes that every constituent computation graph in the fixed choice is an AMM-graph. Since the choice was arbitrary, the stronger claim follows. □

Remark 29 (Scope of MinMax-algebraic RNC state-graph constructions). Definition 236 allows, for every recurrent unit and every relevant time horizon, an arbitrary recurrence graph from the corresponding class $\mathcal{G}_{\mathrm{rec},\circledast}$.

Since $\mathcal{G}_{\mathrm{rec},\circledast}$ permits arbitrary MinMax-algebraic evaluation topology, a graph in $\mathcal{G}_\circledast(D; T, u, x_0)$ may use, in particular, sequential recurrence evaluation, parallel-scan recurrence evaluation, or mixtures of such evaluation topologies across units, layers, and time horizons.

The construction does not permit arbitrary algebraically equivalent smooth reparameterisations of the recurrence. Smooth computation nodes arise only from the smooth graph extensions by the functions $\rho_{\ell,k}$ and g_ℓ prescribed by the RNC architecture. The identity graph $\mathring{I}[\mathring{\theta}]$ introduces no computation

nodes. Apart from these smooth graph extensions, constant parametric graphs, common-input unions, output restrictions, and the graph constructions used to combine them, recurrence evaluation is obtained through MinMax-algebraic extensions and therefore introduces only the scalar minimum and maximum computation nodes prescribed by the MinMax primitive-function family.

The constituent computation graphs satisfying Definition 236 need not be uniquely determined by the terminal graph

$$G \in \mathcal{G}_{\circledast}(D; T, u, x_0)\,.$$

In particular, different choices of recurrence graphs from $\mathcal{G}_{\mathrm{rec},\circledast}$ may yield the same terminal computation graph. All subsequent topology-independent bounds hold for every choice of constituent computation graphs satisfying the defining conditions and hence do not depend on such a choice. No separate construction record is attached to G. *End of Remark 29.*

Theorem 47 (Topology-independent derivative bound for MinMax RNC state evaluation)**.** *Assume the context of Definition 236, let $\theta \in \Theta$, and let*

$$G \in \mathcal{G}_{\circledast}(D; T, u, x_0)\,,$$

Then,

$$\big\|\tilde{\mathsf{D}}_{\circledast} G(\theta)\big\|_{\mathrm{row}} \le \max_{\substack{\ell\in[1..L]\\ k\in[1..n_\ell]\\ t\in[1..T]}} \max\Big\{\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}},\ \big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}}\Big\}\,.$$

Proof. Suppose first that $T = 0$. By Definition 236,

$$G = G_{\mathrm{c}}^{\mathrm{p}}[x_0, \mathring{\theta}].$$

Hence G is parameter-independent and therefore, by Lemma 6,

$$\tilde{\mathsf{D}}_{\circledast} G(\theta) = 0.$$

Suppose now that $T \in \mathbb{N}^{>0}$, and fix any choice of constituent computation graphs satisfying the defining conditions of Definition 236 for G.

By definition,

$$G = \mathring{X}_T^1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_T^L,$$

and, for every $\ell \in [1..L]$,

$$\mathring{X}_T^{\ell} = \mathring{X}_T^{\ell,1} \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_T^{\ell,n_\ell}.$$

By Proposition 48, all these graphs are AMM-graphs, so their $\tilde{\mathsf{D}}_{\circledast}$-pseudo-derivatives are well defined.

Since common-input union concatenates the output node-vectors of its constituent graphs, the rows of $\tilde{\mathsf{D}}_{\circledast} G(\theta)$ are precisely the rows of

$$\tilde{\mathsf{D}}_{\circledast}\mathring{X}_T^{\ell,k}(\theta), \qquad \ell \in [1..L], \quad k \in [1..n_\ell].$$

Hence, by the definition of the row norm,

$$\big\|\tilde{\mathsf{D}}_{\circledast} G(\theta)\big\|_{\mathrm{row}} = \max_{\substack{\ell\in[1..L]\\ k\in[1..n_\ell]}} \big\|\tilde{\mathsf{D}}_{\circledast}\mathring{X}_T^{\ell,k}(\theta)\big\|_{\mathrm{row}}. \tag{101}$$

Fix $\ell \in [1..L]$ and $k \in [1..n_\ell]$. By Definition 236,

$$\mathring{X}_T^{\ell,k} \in \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0^{\ell,k}; \mathring{R}_{[1..T]}^{\ell,k}, \mathring{s}_{[1..T]}^{\ell,k}\big).$$

Therefore, by Theorem 46,

$$\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{X}_T^{\ell,k}(\theta)\big\|_{\mathrm{row}} \le \max_{t\in[1..T]} \max\Big\{\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}},\ \big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}}\Big\}\,.$$

Since ℓ and k were arbitrary, taking the maximum over $\ell \in [1..L]$ and $k \in [1..n_\ell]$, and using (101), yields

$$\big\|\tilde{\mathsf{D}}_{\circledast} G(\theta)\big\|_{\mathrm{row}} \le \max_{\substack{\ell\in[1..L]\\ k\in[1..n_\ell]\\ t\in[1..T]}} \max\Big\{\big\|\tilde{\mathsf{D}}_{\circledast}\mathring{R}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}},\ \big\|\tilde{\mathsf{D}}_{\circledast}\mathring{s}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}}\Big\}\,.$$

□

M.9 Sequential Computation Graphs for MinMax RNCs

Definition 237 (Sequential computation graph)**.** We define the computation graph for evaluating a MinMax recurrence sequentially.

Context. Assume the context of Definition 234.

For every $t \in [1..T]$, let $\mathring{R}_t^{\mathrm{out}}$ and $\mathring{s}_t^{\mathrm{out}}$ be the node-matrix and node-vector output presentations of $\mathring{R}_t$ and $\mathring{s}_t$, respectively, and let $\mathring{x}_0^{\mathrm{out}}$ be the output node-vector of $\mathring{X}_0$. For brevity, in the construction below we write

$$\mathring{x}_0 := \mathring{x}_0^{\mathrm{out}}.$$

Auxiliary. Let us define the common-input union

$$G^{\sqcup} := \mathring{X}_0 \sqcup_{\mathrm{in}} \mathring{R}_1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{R}_T \sqcup_{\mathrm{in}} \mathring{s}_1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{s}_T,$$

regarded as a virtual-computation graph with no virtual nodes.

We set

$$G_0^{\mathrm{seq}} := G^{\sqcup},$$

and, for every $t \in [1..T]$, define the virtual-node extension

$$G_{t-1}^{\mathrm{seq}} \xrightarrow[\mathrm{v}]{\mathring{x}_t := \left(\mathring{R}_t^{\mathrm{out}} \otimes \mathring{x}_{t-1}\right) \oplus \mathring{s}_t^{\mathrm{out}}} G_t^{\mathrm{seq}}. \tag{102}$$

Here the nested MinMax expression is interpreted according to Definition 232; in particular, the required intermediate virtual nodes are introduced as part of the extension.

Main definition: graph instances. The *MinMax sequential virtual-computation graph of horizon T given initial-state graph $\mathring{X}_0$ and reset and set graphs $\mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}$* is the virtual-computation graph

$$\mathrm{seq}_{\circledast,\mathrm{v}}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right) := G_T^{\mathrm{seq}},$$

having output node-vector $\mathring{x}_T$.

The *MinMax sequential computation graph of horizon T given initial-state graph $\mathring{X}_0$ and reset and set graphs $\mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}$*, denoted by

$$\mathrm{seq}_{\circledast}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right),$$

is the computation graph given by the materialisation of

$$\mathrm{seq}_{\circledast,\mathrm{v}}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right)$$

under an arbitrary but fixed materialisation map μ_T^{seq}.

Main definition: graph classes. The *MinMax sequential virtual-computation graph family of horizon T given initial state x_0 and reset and set functions $R_{[1..T]}, s_{[1..T]}$*, denoted by

$$\mathcal{G}_{\circledast}^{\mathrm{seq,v}}(x_0, R_{[1..T]}, s_{[1..T]}),$$

is the set consisting of each graph

$$\mathrm{seq}_{\circledast,\mathrm{v}}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right)$$

for $\mathring{X}_0$ the constant canonical graph for x_0,

$$\mathring{R}_t \in \mathcal{G}_{\circledast}[R_t], \qquad \mathring{s}_t \in \mathcal{G}_{\circledast}[s_t] \qquad \forall t \in [1..T].$$

The *MinMax sequential computation graph family of horizon T given initial state x_0 and reset and set functions $R_{[1..T]}, s_{[1..T]}$*, denoted by

$$\mathcal{G}_{\circledast}^{\mathrm{seq}}(x_0, R_{[1..T]}, s_{[1..T]}),$$

is the set consisting of each graph

$$\mathrm{seq}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

for $\mathring{X}_0$ the constant canonical graph for x_0,

$$\mathring{R}_t \in \mathcal{G}_{\circledast}[R_t]\,, \qquad \mathring{s}_t \in \mathcal{G}_{\circledast}[s_t] \qquad \forall t \in [1..T].$$

End of Definition 237.

Remark 30 (Determination of sequential recurrence graphs). In contrast with the general state-graph construction discussed in Remark 28, the sequential recurrence graph is determined once its source graphs

$$X_0^G, \qquad R_1^G, \ldots, R_T^G, \qquad s_1^G, \ldots, s_T^G$$

are fixed. Indeed, Definition 237 recursively determines the node-vectors

$$\mathring{z}_t = \mathring{R}_t \otimes \mathring{x}_{t-1}, \qquad \mathring{x}_t = \mathring{z}_t \oplus \mathring{s}_t, \qquad t \in [1..T],$$

together with their prescribed scalar expansions.

In particular, $\mathring{x}_1, \ldots, \mathring{x}_{T-1}$ are intermediate node-vectors of the single horizon-T graph G_T^{seq}; they are not independently chosen recurrence graphs of smaller horizons. Thus no additional choice of recurrence topology remains after the source graphs have been fixed. *End of Remark 30.*

Remark 31. In Definition 237, the source graphs

$$\mathring{X}_0, \qquad \mathring{R}_1, \ldots, \mathring{R}_T, \qquad \mathring{s}_1, \ldots, \mathring{s}_T$$

retain all their concrete nodes, multi-edges, and computation-node labels.

Every node added to evaluate the recurrence is a virtual scalar computation node introduced by the MinMax notation of Definition 232. Materialisation replaces these virtual nodes by fresh concrete computation nodes while preserving their predecessor structure, labels, and value functions. *End of Remark 31.*

Proposition 49 (Correctness of the sequential computation graph)**.** *Assume the context of Definition 237.*

For every $\lambda \in \Lambda$ and $t \in [0..T]$,

$$\hat{x}_t(\lambda) = x_t(\lambda). \tag{103}$$

Consequently, for every choice of source graphs admitted by Definition 237, the graph

$$\mathrm{seq}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

represents $\hat{x}_T$.

Proof. We prove Equation (103) by induction on t.

For $t = 0$, the claim follows from the definition of $\mathring{X}_0$:

$$\hat{x}_0(\lambda) = \hat{x}_0^{\mathrm{out}}(\lambda) = x_0 = x_0(\lambda).$$

Let now $t \in [1..T]$ and suppose that

$$\hat{x}_{t-1}(\lambda) = x_{t-1}(\lambda).$$

By Equation (102), Fact 48, and the defining values of the source presentations,

$$\begin{aligned}\hat{x}_t(\lambda) &= \big(\hat{R}_t^{\mathrm{out}}(\lambda) \otimes \hat{x}_{t-1}(\lambda)\big) \oplus \hat{s}_t^{\mathrm{out}}(\lambda) \\ &= \big(R_t(\lambda) \otimes x_{t-1}(\lambda)\big) \oplus s_t(\lambda) \\ &= x_t(\lambda).\end{aligned}$$

This proves Equation (103).

In particular, the final output presentation $\mathring{x}_T$ of

$$\mathrm{seq}_{\circledast,\mathrm{v}}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

represents x_T. By Proposition 21, materialisation preserves its value function, and hence the value of the materialised graph

$$\mathrm{seq}_{\circledast}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right)$$

is x_T. □

Proposition 50 (Sequential graphs are MinMax-algebraic)**.** *Assume the context of Definition 237.*

For every choice of source graphs admitted by that definition,

$$\mathrm{seq}_{\circledast,\mathrm{v}}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right) \in \mathcal{E}^{\mathrm{v}}_{\circledast}[\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T], \tag{104}$$

and

$$\mathrm{seq}_{\circledast}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right) \in \mathcal{E}_{\circledast}\left(\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T\right). \tag{105}$$

Consequently,

$$\mathcal{G}^{\mathrm{seq}}_{\circledast}(x_0, R_{[1..T]}, s_{[1..T]}) \subseteq \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; R_{[1..T]}, s_{[1..T]}\big). \tag{106}$$

Proof. By construction,

$$\mathrm{seq}_{\circledast,\mathrm{v}}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right)$$

is a virtual-node extension of the common-input union $G^{\sqcup}$ of

$$\mathring{X}_0, \qquad \mathring{R}_1, \ldots, \mathring{R}_T, \qquad \mathring{s}_1, \ldots, \mathring{s}_T.$$

Every extension in Equation (102) is specified by the MinMax notation of Definition 232. Hence every scalar virtual computation node introduced by the sequential recurrence construction has a label admitted by Definition 230. It follows that

$$\mathrm{seq}_{\circledast,\mathrm{v}}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right) \in \mathcal{E}^{\mathrm{v}}_{\circledast}[\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T],$$

proving Equation (104).

By Proposition 44, its materialisation belongs to the corresponding ordinary MinMax-algebraic extension class. Thus

$$\mathrm{seq}_{\circledast}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right) \in \mathcal{E}_{\circledast}\left(\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T\right),$$

proving Equation (105).

Proposition 49 gives that the graph

$$\mathrm{seq}_{\circledast}\left(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\right)$$

represents $\hat{x}_T$. Therefore, by Definition 234, every such materialised sequential graph belongs to

$$\mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; R_{[1..T]}, s_{[1..T]}\big).$$

Since the source graphs were arbitrary members of the graph classes specified in Definition 237, Equation (106) follows. □

M.10 Parallel-Scan Computation Graphs for MinMax RNCs

Definition 238 (Parallel-scan computation graph)**.** We define the computation graph for evaluating MinMax recurrence via *exclusive parallel-scan*.

Context. Assume the context of Definition 234.

For every $t \in [1..T]$, let $\mathring{R}^{\mathrm{out}}_t$ and $\mathring{s}^{\mathrm{out}}_t$ be the node-matrix and node-vector output presentations of $\mathring{R}_t$ and $\mathring{s}_t$, respectively, and let $\mathring{x}^{\mathrm{out}}_0$ be the output node-vector of $\mathring{X}_0$. For some $n \in \mathbb{N}^{>0}$, we have that each $\mathring{R}^{\mathrm{out}}_t$ has dimension n^2, and each $\mathring{s}^{\mathrm{out}}_t$ has dimension n.

Auxiliary. Let us define the common-input union

$$G^{\sqcup} := \mathring{X}_0 \sqcup_{\mathrm{in}} \mathring{R}_1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{R}_T \sqcup_{\mathrm{in}} \mathring{s}_1 \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{s}_T$$

regarded as a virtual-computation graph with no virtual nodes.

Let $\mathring{\mathcal{V}}$ be the node-vector obtained by concatenating all scalar entries of

$$\mathring{x}_0^{\mathrm{out}}, \quad \mathring{R}_1^{\mathrm{out}}, \ldots, \mathring{R}_T^{\mathrm{out}}, \quad \mathring{s}_1^{\mathrm{out}}, \ldots, \mathring{s}_T^{\mathrm{out}}.$$

We extend $G^{\sqcup}$ as follows:

$$G^{\sqcup} \xrightarrow[\mathrm{v}]{\mathring{v}_{\mathsf{min}} := \mathsf{min}(\mathring{\mathcal{V}}),\ \mathring{v}_{\mathsf{max}} := \mathsf{max}(\mathring{\mathcal{V}}),\ \mathring{E},\ \mathring{e},\ \mathring{M}_{\mathrm{id}} := (\mathring{E}, \mathring{e})} G_0^{\mathrm{src}}$$

where the node-matrix $\mathring{E}$ and the node-vector $\mathring{e}$ have components as follows:

$$(\mathring{E})_{i,j} := \begin{cases} \mathring{v}_{\mathsf{max}} & \text{if } i = j, \\ \mathring{v}_{\mathsf{min}} & \text{if } i \neq j, \end{cases} \qquad (\mathring{e})_i := \mathring{v}_{\mathsf{min}}, \qquad \forall i, j \in [1..n].$$

Then we define the extensions

$$G_{t-1}^{\mathrm{src}} \xrightarrow[\mathrm{v}]{\mathring{M}_t := (\mathring{R}_t^{\mathrm{out}}, \mathring{s}_t^{\mathrm{out}})} G_t^{\mathrm{src}} \qquad \forall t \in [1..T].$$

Exclusive-scan virtual-computation graph. For every $t \in \mathbb{N}^{>0}$, we inductively define the *exclusive-scan virtual-computation graph*

$$\mathrm{xscan}_t\big(\mathring{M}_{\mathrm{id}}; \mathring{M}_{[1..t]} \,|\, G_t^{\mathrm{src}}\big) \quad \text{having output node-vector} \quad \big(\mathring{P}_1, \ldots, \mathring{P}_t\big),$$

as follows:

- when $t = 1$, the exclusive-scan virtual-computation graph is given by the following extension:

$$G_1^{\mathrm{src}} \xrightarrow[\mathrm{v}]{\mathring{P}_1 := \mathring{M}_{\mathrm{id}}} \mathrm{xscan}_t\big(\mathring{M}_{\mathrm{id}}; \mathring{M}_{[1..t]} \,|\, G_t^{\mathrm{src}}\big)$$

- when $t > 1$, we define $m := \lceil \frac{t}{2} \rceil$, and then we proceed as follows:
  - we first define the extension

$$\begin{cases} G_t^{\mathrm{src}} \xrightarrow[\mathrm{v}]{\big(\mathring{C}_\tau := \mathring{M}_{2\tau-1} \circledast \mathring{M}_{2\tau} : \tau \in [1..t/2]\big)} G_t^{\mathrm{c}} & \text{if } t \text{ is even}, \\ G_t^{\mathrm{src}} \xrightarrow[\mathrm{v}]{\big(\mathring{C}_\tau := \mathring{M}_{2\tau-1} \circledast \mathring{M}_{2\tau} : \tau \in [1..\lfloor t/2 \rfloor]\big),\ \mathring{C}_m := \mathring{M}_t} G_t^{\mathrm{c}} & \text{if } t \text{ is odd}; \end{cases}$$

  - the induction yields the exclusive-scan virtual-computation graph for $m < t$,

$$G_m^{\mathrm{rec}} := \mathrm{scan}_m\big(\mathring{M}_{\mathrm{id}}; \mathring{C}_1, \ldots, \mathring{C}_m \;\big|\; G_t^{\mathrm{c}}\big), \quad \text{with output node-vector } \big(\mathring{Q}_1, \ldots, \mathring{Q}_m\big),$$

  - then, we define the extension

$$\begin{cases} G_t^{\mathrm{rec}} \xrightarrow[\mathrm{v}]{\big(\mathring{P}_{2\tau-1} := \mathring{Q}_\tau,\ \mathring{P}_{2\tau} := \mathring{Q}_\tau \circledast \mathring{M}_{2\tau-1} : \tau \in [1..t/2]\big)} G_t^{\mathrm{scan}} & \text{if } t \text{ is even}, \\ G_t^{\mathrm{rec}} \xrightarrow[\mathrm{v}]{\big(\mathring{P}_{2\tau-1} := \mathring{Q}_\tau,\ \mathring{P}_{2\tau} := \mathring{Q}_\tau \circledast \mathring{M}_{2\tau-1} : \tau \in [1..\lfloor t/2 \rfloor]\big),\ \mathring{P}_t := \mathring{Q}_m} G_t^{\mathrm{scan}} & \text{if } t \text{ is odd}; \end{cases}$$

- finally, we define the exclusive-scan virtual-computation graph as the graph

$$\mathrm{xscan}_t\big(\mathring{M}_{\mathrm{id}}; \mathring{M}_{[1..t]} \,|\, G_t^{\mathrm{src}}\big) := G_t^{\mathrm{scan}}.$$

Inclusive virtual-computation graphs. We extend the exclusive scan graphs to compute states:

$$\mathrm{xscan}_t\big(\mathring{M}_{\mathrm{id}};\mathring{M}_{[1..t]}\,|\,G_t^{\mathrm{src}}\big)\xRightarrow[\mathrm{v}]{(\mathring{A}_{1:t},\mathring{b}_{1:t}):=\mathring{P}_t\circledast\mathring{M}_t,\ \ \mathring{x}_t:=(\mathring{A}_{1:t}\otimes\mathring{x}_0)\oplus\mathring{b}_{1:t}}\mathrm{scan}_t\big(\mathring{M}_{\mathrm{id}};\mathring{M}_{[1..t]}\,|\,G_t^{\mathrm{src}}\big)$$

for every $t\in[1..T]$.

Main definition: graph instances. The *MinMax parallel-scan virtual-computation graph of horizon T given initial state graph $\mathring{X}_0$ and reset and set graphs $\mathring{R}_{[1..T]},\mathring{s}_{[1..T]}$* is the virtual-computation graph

$$\mathrm{scan}_{\circledast,\mathrm{v}}\big(\mathring{X}_0,\mathring{R}_{[1..T]},\mathring{s}_{[1..T]},\big):=\mathrm{scan}_t\big(\mathring{M}_{\mathrm{id}};\mathring{M}_{[1..t]}\,|\,G_t^{\mathrm{src}}\big)$$

having output node-vector $\mathring{x}_T$.

The *MinMax parallel-scan computation graph of horizon T given initial state graph $\mathring{X}_0$ and reset and set graphs $\mathring{R}_{[1..T]},\mathring{s}_{[1..T]}$* denoted by

$$\mathrm{scan}_{\circledast}\big(\mathring{X}_0,\mathring{R}_{[1..T]},\mathring{s}_{[1..T]},\big)$$

is the computation graph given by the materialisation of

$$\mathrm{scan}_{\circledast,\mathrm{v}}\big(\mathring{X}_0,\mathring{R}_{[1..T]},\mathring{s}_{[1..T]},\big)$$

under an arbitrary but fixed materialisation map μ_T^{scan}.

Main definition: graph classes. The *MinMax parallel-scan virtual-computation graph family of horizon T given initial state x_0 and reset and set functions $R_{[1..T]},s_{[1..T]}$* denoted by

$$\mathcal{G}_{\circledast}^{\mathrm{scan,v}}\big(x_0,R_{[1..T]},s_{[1..T]},\big)$$

is the set consisting of each graph

$$\mathrm{scan}_{\circledast,\mathrm{v}}\big(\mathring{X}_0,\mathring{R}_{[1..T]},\mathring{s}_{[1..T]},\big)$$

for $\mathring{X}_0$ the constant canonical graph for x_0, $\mathring{R}_t\in\mathcal{G}_{\circledast}[R_t]$, and $\mathring{s}_t\in\mathcal{G}_{\circledast}[s_t]$.

The *MinMax parallel-scan computation graph family of horizon T given initial state x_0 and reset and set functions $R_{[1..T]},s_{[1..T]}$*, denoted by

$$\mathcal{G}_{\circledast}^{\mathrm{scan}}\big(x_0,R_{[1..T]},s_{[1..T]},\big)$$

is the set consisting of each graph

$$\mathrm{scan}_{\circledast}\big(\mathring{X}_0,\mathring{R}_{[1..T]},\mathring{s}_{[1..T]},\big)$$

for $\mathring{X}_0$ the constant canonical graph for x_0, $\mathring{R}_t\in\mathcal{G}_{\circledast}[R_t]$, and $\mathring{s}_t\in\mathcal{G}_{\circledast}[s_t]$. *End of Definition 238.*

Remark 32 (Determination of parallel-scan recurrence graphs). As in the sequential case, and in contrast with the general state-graph construction of Remark 28, the parallel-scan construction determines its recurrence graph once the source graphs

$$X_0^G,\qquad R_1^G,\dots,R_T^G,\qquad s_1^G,\dots,s_T^G$$

and the materialisation map used in Definition 238 are fixed.

In particular, the presentations $\mathring{x}_t$, for $t\in[1..T]$, are determined within the single parallel-scan construction and are not independently chosen recurrence graphs of the corresponding horizons. Hence, after fixing the source graphs and the materialisation, no additional choice of recurrence topology remains. *End of Remark 32.*

Fact 51 (Values and data-generated closure of the parallel scan)**.** *Assume the context of Definition 238.*

For every $\lambda\in\Lambda$, let

$$\mathcal{A}_\lambda:=V_{\circledast,\lambda}^{n\times n}\times V_{\circledast,\lambda}^{n}.$$

For every $t\in[1..T]$,

$$\hat{x}_0^{\mathrm{out}}(\lambda)=x_0,\qquad \hat{R}_t^{\mathrm{out}}(\lambda)=R_t(\lambda),\qquad \hat{s}_t^{\mathrm{out}}(\lambda)=s_t(\lambda).\tag{107}$$

Furthermore,

$$\hat{v}_{\mathsf{min}}(\lambda) = v_{\min,\lambda}, \qquad \hat{v}_{\mathsf{max}}(\lambda) = v_{\max,\lambda}. \tag{108}$$

The node-matrix and node-vector presentations $\mathring{E}$ and $\mathring{e}$ represent parametric functions

$$E : \Lambda \to \mathbb{R}^{n\times n}, \qquad e : \Lambda \to \mathbb{R}^{n}$$

given by

$$(E(\lambda))_{i,j} = \begin{cases} v_{\max,\lambda}, & \text{if } i = j, \\ v_{\min,\lambda}, & \text{if } i \neq j, \end{cases} \qquad (e(\lambda))_i = v_{\min,\lambda}. \tag{109}$$

Hence $\mathring{M}_{\mathrm{id}}$ represents

$$M_{\mathrm{id}} : \Lambda \to \mathbb{R}^{n\times n} \times \mathbb{R}^{n}, \qquad M_{\mathrm{id}}(\lambda) := (E(\lambda), e(\lambda)).$$

Furthermore, $M_{\mathrm{id}}(\lambda)$ is the identity element under $\circledast$ on $\mathcal{A}_\lambda$.

For every $t \in [1..T]$,

$$\hat{M}_t(\lambda) = M_t(\lambda) := (R_t(\lambda), s_t(\lambda)) \in \mathcal{A}_\lambda. \tag{110}$$

Finally, $\mathcal{A}_\lambda$ is closed under $\circledast$. Consequently, every MinMax affine operator represented by a node presentation generated during the recursive exclusive scan or the subsequent inclusive step belongs to $\mathcal{A}_\lambda$.

Proof. Equation (107) follows from the definitions of $\mathring{X}_0$, $\mathring{R}_t$, and $\mathring{s}_t$.

By Equation (107), the values at λ of the entries of $\mathring{\mathcal{V}}$ are precisely the elements used to generate $V_{\circledast,\lambda}$. Therefore, by the minimum- and maximum-reduction clauses of Definition 232 and Fact 48,

$$\hat{v}_{\mathsf{min}}(\lambda) = v_{\min,\lambda}, \qquad \hat{v}_{\mathsf{max}}(\lambda) = v_{\max,\lambda},$$

proving Equation (108).

By the definitions of the components of $\mathring{E}$ and $\mathring{e}$, together with the copy clauses of Definition 232 and Fact 48, Equation (108) gives Equation (109). The same copy-value property applied to

$$\mathring{M}_{\mathrm{id}} := (\mathring{E}, \mathring{e})$$

gives

$$\hat{M}_{\mathrm{id}}(\lambda) = (E(\lambda), e(\lambda)) = M_{\mathrm{id}}(\lambda).$$

Proposition 19 then implies that $M_{\mathrm{id}}(\lambda)$ is the identity element under $\circledast$ on $\mathcal{A}_\lambda$.

For every $t \in [1..T]$, the copy-value property applied to

$$\mathring{M}_t := (\mathring{R}_t^{\mathrm{out}}, \mathring{s}_t^{\mathrm{out}})$$

and Equation (107) gives

$$\hat{M}_t(\lambda) = (R_t(\lambda), s_t(\lambda)).$$

By definition of $V_{\circledast,\lambda}$, every entry of $R_t(\lambda)$ and $s_t(\lambda)$ belongs to $V_{\circledast,\lambda}$. Hence Equation (110) follows.

Finally, Proposition 18 implies that $\mathcal{A}_\lambda$ is closed under $\circledast$. Since $M_{\mathrm{id}}(\lambda)$ and every $M_t(\lambda)$ belong to $\mathcal{A}_\lambda$, every MinMax affine operator obtained from them by the $\circledast$ operations appearing in the recursive exclusive scan and in the inclusive step also belongs to $\mathcal{A}_\lambda$. □

Proposition 51 (Correctness of the parallel-scan computation graph)**.** *Assume the context of Definition 238.*

For every $\lambda \in \Lambda$ and $t \in [1..T]$, let

$$M_t(\lambda) := (R_t(\lambda), s_t(\lambda)),$$

and let

$$M_{1:t}(\lambda) := M_1(\lambda) \circledast \cdots \circledast M_t(\lambda).$$

Then the exclusive-scan presentation $\mathring{P}_t$ satisfies

$$\hat{P}_t(\lambda) = \begin{cases} M_{\mathrm{id}}(\lambda), & t = 1, \\ M_{1:t-1}(\lambda), & t > 1. \end{cases} \tag{111}$$

Consequently, the inclusive presentations introduced by scan_t *satisfy*

$$(\hat{A}_{1:t}(\lambda), \hat{b}_{1:t}(\lambda)) = M_{1:t}(\lambda). \tag{112}$$

Writing

$$M_{1:t}(\lambda) = (A_{1:t}(\lambda), b_{1:t}(\lambda)),$$

we furthermore have

$$\hat{x}_t(\lambda) = \big(A_{1:t}(\lambda) \otimes x_0\big) \oplus b_{1:t}(\lambda) = x_t(\lambda). \tag{113}$$

In particular, for every choice of source graphs as in Definition 238, the graph

$$\mathrm{scan}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

represents $\hat{x}_T$.

Proof. Fix $\lambda \in \Lambda$ and let

$$\mathcal{A}_\lambda := V_{\circledast,\lambda}^{n\times n} \times V_{\circledast,\lambda}^{n}.$$

By Fact 51,

$$\hat{M}_t(\lambda) = M_t(\lambda) \in \mathcal{A}_\lambda \qquad \forall t \in [1..T], \tag{114}$$

$M_{\mathrm{id}}(\lambda)$ is the identity element under $\circledast$ on $\mathcal{A}_\lambda$, and $\mathcal{A}_\lambda$ is closed under $\circledast$.

We first establish correctness of the recursive exclusive scan. More generally, consider an invocation

$$\mathrm{xscan}_m\big(\mathring{M}_{\mathrm{id}}; \mathring{N}_1, \dots, \mathring{N}_m \;\big|\; H\big),$$

where

$$\hat{N}_1(\lambda) = N_1, \dots, \hat{N}_m(\lambda) = N_m \in \mathcal{A}_\lambda,$$

and let $\mathring{P}_1, \dots, \mathring{P}_m$ be its exclusive-scan presentations. We claim that

$$\hat{P}_r(\lambda) = \begin{cases} M_{\mathrm{id}}(\lambda), & r = 1, \\ N_1 \circledast \cdots \circledast N_{r-1}, & r > 1, \end{cases} \qquad \forall r \in [1..m]. \tag{115}$$

We prove the claim by induction on m. If $m = 1$, the base-case extension gives

$$\mathring{P}_1 := \mathring{M}_{\mathrm{id}}.$$

Therefore, by Fact 48,

$$\hat{P}_1(\lambda) = M_{\mathrm{id}}(\lambda).$$

Suppose now that $m > 1$ and that the claim holds for every smaller scan length. Let

$$\tilde{m} := \left\lceil \frac{m}{2} \right\rceil.$$

For every $j \in [1..\lfloor m/2 \rfloor]$, the contraction step introduces

$$\mathring{C}_j := \mathring{N}_{2j-1} \circledast \mathring{N}_{2j}.$$

By Fact 48, it represents

$$C_j := N_{2j-1} \circledast N_{2j}. \tag{116}$$

If m is odd, then the contraction step additionally introduces

$$\mathring{C}_{\tilde{m}} := \mathring{N}_m,$$

and therefore

$$C_{\tilde{m}} = N_m.$$

By closure of $\mathcal{A}_\lambda$ under $\circledast$, every C_j belongs to $\mathcal{A}_\lambda$. Since $\tilde{m} < m$, the induction hypothesis applies to the recursive invocation

$$\mathrm{xscan}_{\tilde{m}}\big(\mathring{M}_{\mathrm{id}}; \mathring{C}_1, \ldots, \mathring{C}_{\tilde{m}} \;\big|\; G_m^{\mathrm{c}}\big).$$

Its exclusive-scan presentations $\mathring{Q}_1, \ldots, \mathring{Q}_{\tilde{m}}$ therefore satisfy

$$\hat{Q}_j(\lambda) = \begin{cases} M_{\mathrm{id}}(\lambda), & j = 1, \\ C_1 \circledast \cdots \circledast C_{j-1}, & j > 1. \end{cases} \tag{117}$$

By Equation (116) and associativity of $\circledast$, for every $j > 1$,

$$\hat{Q}_j(\lambda) = N_1 \circledast \cdots \circledast N_{2j-2}. \tag{118}$$

For every $j \in [1..\lfloor m/2 \rfloor]$, the expansion step introduces

$$\mathring{P}_{2j-1} := \mathring{Q}_j.$$

Hence, by the copy-value property, if $j = 1$,

$$\hat{P}_1(\lambda) = M_{\mathrm{id}}(\lambda),$$

while, for $j > 1$,

$$\hat{P}_{2j-1}(\lambda) = N_1 \circledast \cdots \circledast N_{2j-2}.$$

Furthermore,

$$\mathring{P}_{2j} := \mathring{Q}_j \circledast \mathring{N}_{2j-1}.$$

Therefore, by Fact 48, Equation (118) for $j > 1$, and the identity property for $j = 1$,

$$\hat{P}_{2j}(\lambda) = N_1 \circledast \cdots \circledast N_{2j-1}.$$

Finally, if m is odd, then

$$m = 2\tilde{m} - 1, \qquad \mathring{P}_m := \mathring{Q}_{\tilde{m}}.$$

Hence, by the copy-value property and Equation (118),

$$\hat{P}_m(\lambda) = N_1 \circledast \cdots \circledast N_{m-1}.$$

This proves Equation (115).

Applying the general claim to

$$\mathring{N}_t = \mathring{M}_t, \qquad N_t = M_t(\lambda), \qquad t \in [1..T],$$

gives Equation (111).

By the definition of the inclusive extension,

$$(\mathring{A}_{1:t}, \mathring{b}_{1:t}) := \mathring{P}_t \circledast \mathring{M}_t.$$

Hence, by Fact 48, for $t = 1$,

$$\begin{aligned} (\hat{A}_{1:1}(\lambda), \hat{b}_{1:1}(\lambda)) &= M_{\mathrm{id}}(\lambda) \circledast M_1(\lambda) \\ &= M_1(\lambda), \end{aligned}$$

while, for $t > 1$,

$$\begin{aligned} (\hat{A}_{1:t}(\lambda), \hat{b}_{1:t}(\lambda)) &= M_{1:t-1}(\lambda) \circledast M_t(\lambda) \\ &= M_{1:t}(\lambda). \end{aligned}$$

This proves Equation (112).

Write

$$M_{1:t}(\lambda) = (A_{1:t}(\lambda), b_{1:t}(\lambda)).$$

By Definition 211 and associativity of $\circledast$, $M_{1:t}(\lambda)$ represents the composition, in temporal order, of the MinMax affine maps

$$x \longmapsto \big(R_\tau(\lambda) \otimes x\big) \oplus s_\tau(\lambda), \qquad \tau \in [1..t].$$

Applying this composed operator to x_0 yields

$$x_t(\lambda) = \big(A_{1:t}(\lambda) \otimes x_0\big) \oplus b_{1:t}(\lambda).$$

By the definition of $\mathring{x}_t$, Equation (107), Equation (112), and Fact 48, the right-hand side equals $\hat{x}_t(\lambda)$. This proves Equation (113).

In particular, the final output presentation $\mathring{x}_T$ of

$$\mathrm{scan}_{\circledast,\mathrm{v}}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

represents x_T. By Proposition 44, materialisation preserves its value function, and hence the value of the materialised graph

$$\mathrm{scan}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

is x_T. □

Proposition 52 (Parallel-scan graphs are MinMax-algebraic)**.** *Assume the context of Definition 238.*

For every choice of

$$\mathring{X}_0, \qquad \mathring{R}_{[1..T]}, \qquad \mathring{s}_{[1..T]}$$

admitted by that definition,

$$\mathrm{scan}_{\circledast,\mathrm{v}}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big) \in \mathcal{E}^{\mathrm{v}}_{\circledast}[\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T], \tag{119}$$

and

$$\mathrm{scan}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big) \in \mathcal{E}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T\Big). \tag{120}$$

Consequently,

$$\mathcal{G}^{\mathrm{scan}}_{\circledast}(x_0, R_{[1..T]}, s_{[1..T]}) \subseteq \mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; R_{[1..T]}, s_{[1..T]}\big). \tag{121}$$

Proof. By construction,

$$\mathrm{scan}_{\circledast,\mathrm{v}}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

is a virtual-node extension of the common-input union $G^{\sqcup}$ of

$$\mathring{X}_0, \qquad \mathring{R}_1, \ldots, \mathring{R}_T, \qquad \mathring{s}_1, \ldots, \mathring{s}_T.$$

The virtual nodes $\mathring{v}_{\mathsf{min}}$ and $\mathring{v}_{\mathsf{max}}$ are labelled by

$$\mathsf{min}_{|\mathring{\mathcal{V}}|} \qquad \text{and} \qquad \mathsf{max}_{|\mathring{\mathcal{V}}|},$$

respectively.

The scalar virtual nodes introduced by the node-matrix, node-vector, and tuple-copy presentations

$$\mathring{E}, \qquad \mathring{e}, \qquad \mathring{M}_{\mathrm{id}}, \qquad \mathring{M}_1, \ldots, \mathring{M}_T$$

are identity-labelled virtual computation nodes, as prescribed by Definition 232. These labels are admitted by Definition 230.

Every occurrence of $\circledast$ in the contraction, recursive expansion, and inclusive stages is expanded, by Definition 211, into MinMax matrix and vector expressions. By Definition 232, every scalar virtual computation node introduced by these expressions is labelled by an identity, minimum, or maximum operation admitted by Definition 230. The same holds for the final state presentations $\mathring{x}_t$, for $t \in [1..T]$.

Therefore,

$$\mathrm{scan}_{\circledast,\mathrm{v}}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big) \in \mathcal{E}^{\mathrm{v}}_{\circledast}[\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T],$$

proving Equation (119).

By Proposition 44, every materialisation of this graph belongs to the corresponding ordinary MinMax-algebraic extension class. In particular,

$$\mathrm{scan}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big) \in \mathcal{E}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_1, \ldots, \mathring{R}_T, \mathring{s}_1, \ldots, \mathring{s}_T\Big),$$

proving Equation (120).

Proposition 51 gives that the graph

$$\mathrm{scan}_{\circledast}\Big(\mathring{X}_0, \mathring{R}_{[1..T]}, \mathring{s}_{[1..T]}\Big)$$

represents $\hat{x}_T$. Therefore, by Definition 234, every such materialised parallel-scan graph belongs to

$$\mathcal{G}_{\mathrm{rec},\circledast}\big(x_0; R_{[1..T]}, s_{[1..T]}\big).$$

Since the source graphs were arbitrary members of the graph classes specified in Definition 238, Equation (121) follows. □

M.11 BIBSD: Uniform Selected uBIBSD for MinMax RNCs

We now lift the topology-independent AMM-pseudo-derivative bound of Theorem 47 from individual MinMax recurrence graphs to complete MinMax RNC dynamics.

The required uniform state-value bound is already provided by Corollary 20. Thus, no additional stability argument is needed in this section: the only remaining task is to propagate bounded values and bounded selected derivatives through the finitely many analytic maps occurring between recurrent layers.

Context 1. *For this section, we consider arbitrary but fixed parametric MinMax RNC dynamics*

$$D \in \mathrm{RNC}_{\mathrm{dyn}}\left(\mathcal{U}^{\mathrm{p}}_{\mathrm{AN}}[\mathrm{MinMax}]\right), \qquad D = \langle X, U, f; \Theta\rangle,$$

and we write $\mathcal{G}_{\circledast}$ for the corresponding state computation-graph family of D as per Definition 236. We note that the class $\mathcal{G}_{\circledast}$ is an AMM-computation-graph family, by Proposition 48.

Remark 33 (Graph choices in the MinMax BIBSD result). The BIBSD result below is stated for the general state computation-graph class $\mathcal{G}_{\circledast}$, which provides great flexibility in choosing the subgraphs of a state computation graph—as discussed in Remark 28. The bound below is uniform over all such admissible choices and does not depend on selecting a particular recurrence-evaluation topology.

The sequential and parallel-scan constructions yield specific graphs included in $\mathcal{G}_{\circledast}$. For such graphs, once the corresponding source graphs are fixed, the recurrence computation is determined as described in Remarks 30 and 32, respectively. *End of Remark 33.*

Theorem 48 (Graph-uniform selected uBIBSD of parametric MinMax RNC dynamics)**.** *Parametric MinMax RNC dynamics D satisfy $\mathcal{G}_{\circledast}$-$\tilde{\mathsf{D}}_{\circledast}$–uBIBSD. Specifically, for every $r \in \mathbb{R}^{\geq 0}$, there exists $B_r \in \mathbb{R}^{\geq 0}$ such that*

$$\big\|\tilde{\mathsf{D}}_{\circledast} G(\theta)\big\|_{\mathrm{row}} \leq B_r \qquad \forall u \in U^{\omega}_{\leq r},\ \forall \dot{x} \in X_{\leq r},\ \forall T \in \mathbb{T},\ \forall \theta \in \Theta_{\leq r},\ \forall G \in \mathcal{G}_{\circledast}(D; T, u, \dot{x})\ .$$

Remark 34. The bounding constant B_r is independent of u, $\dot{x}$, T, and of the graph G—in particular, it is independent of its MinMax-algebraic recurrence-evaluation topology and the chosen constituent computation-graphs. *End of Remark 34.*

Proof. Let $r \in \mathbb{R}^{\geq 0}$ be arbitrary. By Corollary 20, D satisfies uBIBS. Hence there exists $B^X_r \in \mathbb{R}^{\geq 0}$ such that, for all $u \in U^{\omega}_{\leq r}$, $\dot{x} \in X_{\leq r}$, $\theta \in \Theta_{\leq r}$, the corresponding state sequence $(x_t)_{t\in\mathbb{T}}$ satisfies

$$\sup_{t\in\mathbb{T}} \|x_t\| \leq B^X_r. \tag{122}$$

Since the RNC has finitely many layers and every layer-state space is finite-dimensional, after enlarging B^X_r by a fixed norm-equivalence factor if necessary, we use the same symbol for a uniform bound on every layer-state block.

Let us fix an abitrary input sequence $u \in U^{\omega}_{\leq r}$, initial state $\dot{x} \in X_{\leq r}$, horizon $T \in \mathbb{T}$, and computation graph $G \in \mathcal{G}_{\circledast}(D; T, u, \dot{x})$.

By Proposition 48, G is an AMM-graph. Hence Proposition 43 implies that $\tilde{\mathsf{D}}_{\circledast}$ is intensionally-admissible for G. Since G was arbitrary, $\tilde{\mathsf{D}}_{\circledast}$ is intensionally-admissible for the complete family $\mathcal{G}_{\circledast}$.

If $T = 0$, then $G = G^{\mathrm{p}}_{\mathrm{c}}[\dot{x}, \mathring{\theta}]$, and therefore $\tilde{\mathsf{D}}_{\circledast} G(\theta) = 0$. Thus the claim is immediate.

Hence, suppose now that $T \in \mathbb{N}^{>0}$. Fix arbitrary constituent computation graphs

$$\mathring{U}^{\ell}_t, \quad \mathring{R}^{\ell,k}_t, \quad \mathring{s}^{\ell,k}_t, \quad \mathring{H}^{\ell,k}_t, \quad \mathring{X}^{\ell,k}_t, \quad \mathring{X}^{\ell}_t$$

satisfying Definition 236 for G.

We prove inductively over $\ell \in [1..L]$ that there exist constants

$$B^{U}_{\ell}, \qquad B^{U'}_{\ell}, \qquad M_{\ell} \in \mathbb{R}^{\geq 0}$$

depending only on r and the fixed RNC architecture such that the following bounds hold uniformly over u, $\dot{x}$, T, $\theta \in \Theta_{\leq r}$, G, the constituent graph choices, and the relevant time indices:

$$\left\|\hat{U}^{\ell}_t(\theta)\right\| \leq B^{U}_{\ell}, \qquad \left\|\tilde{\mathsf{D}}_{\circledast} \mathring{U}^{\ell}_t(\theta)\right\|_{\mathrm{row}} \leq B^{U'}_{\ell}, \tag{123}$$

$$\max\left\{\left\|\tilde{\mathsf{D}}_{\circledast} \mathring{R}^{j,k}_t(\theta)\right\|_{\mathrm{row}}, \ \left\|\tilde{\mathsf{D}}_{\circledast} \mathring{s}^{j,k}_t(\theta)\right\|_{\mathrm{row}}\right\} \leq M_{\ell} \qquad \forall j \in [1..\ell]. \tag{124}$$

For the first layer, Definition 236 gives

$$\mathring{U}^{1}_t = G^{\mathrm{p}}_{\mathrm{c}}[u_t, \mathring{\theta}].$$

Therefore

$$\left\|\hat{U}^{1}_t(\theta)\right\| = \|u_t\| \leq r, \qquad \tilde{\mathsf{D}}_{\circledast} \mathring{U}^{1}_t(\theta) = 0.$$

Hence we may take

$$B^{U}_{1} = r, \qquad B^{U'}_{1} = 0.$$

The identity parameter graph satisfies

$$\hat{I}[\mathring{\theta}](\theta) = \theta, \qquad \tilde{\mathsf{D}}_{\circledast} \mathring{I}[\mathring{\theta}](\theta) = I_{d\Theta}.$$

Consequently, on $\Theta_{\leq r}$, both its value and its selected derivative are uniformly bounded.

Let now $\ell \in [1..L]$ and suppose that Equation (123) holds at layer ℓ, and, when $\ell > 1$, that Equation (124) holds through layer $\ell - 1$.

Fix $k \in [1..n_{\ell}]$. By Definition 236,

$$\mathring{H}^{\ell,k}_t = (\mathring{U}^{\ell}_t, \mathring{I}[\mathring{\theta}]) \triangleright \rho_{\ell,k},$$

where $\rho_{\ell,k}$ is analytic.

The families of source graphs

$$\left\{\mathring{U}^{\ell}_t\right\}, \qquad \left\{\mathring{I}[\mathring{\theta}]\right\}$$

have uniformly bounded values and uniformly bounded selected derivatives. Therefore, by Lemma 13, the graphs $\mathring{H}^{\ell,k}_t$ have uniformly bounded values and uniformly bounded AMM-pseudo-derivatives.

The reset and set graphs are output restrictions:

$$\mathring{R}^{\ell,k}_t = \left(\mathring{H}^{\ell,k}_t\right)_{J^{\ell,k}_R}, \qquad \mathring{s}^{\ell,k}_t = \left(\mathring{H}^{\ell,k}_t\right)_{J^{\ell,k}_s}.$$

Output restriction only removes rows of the selected Jacobian and therefore cannot increase the row norm. Since layer ℓ contains finitely many units, there exists $\hat{M}_{\ell} \in \mathbb{R}^{\geq 0}$ such that

$$\max\left\{\left\|\tilde{\mathsf{D}}_{\circledast} \mathring{R}^{\ell,k}_t(\theta)\right\|_{\mathrm{row}}, \ \left\|\tilde{\mathsf{D}}_{\circledast} \mathring{s}^{\ell,k}_t(\theta)\right\|_{\mathrm{row}}\right\} \leq \hat{M}_{\ell} \tag{125}$$

uniformly over k, t, and all admissible choices.

Define

$$M_\ell := \begin{cases} \hat{M}_1, & \ell = 1, \\ \max\left\{M_{\ell-1}, \hat{M}_\ell\right\}, & \ell > 1. \end{cases}$$

Then Equation (124) holds through layer ℓ.

For every $t \in [1..T]$ and every $k \in [1..n_\ell]$, Definition 236 gives

$$\mathring{X}_t^{\ell,k} \in \mathcal{G}_{\mathrm{rec},\circledast}\big(\dot{x}^{\ell,k}; \mathring{R}_{[1..t]}^{\ell,k}, \mathring{s}_{[1..t]}^{\ell,k}\big).$$

Therefore, Theorem 46 and Equation (125) yield

$$\big\|\tilde{\mathsf{D}}_\circledast \mathring{X}_t^{\ell,k}(\theta)\big\|_{\mathrm{row}} \le \hat{M}_\ell \le M_\ell. \tag{126}$$

Since

$$\mathring{X}_t^{\ell} = \mathring{X}_t^{\ell,1} \sqcup_{\mathrm{in}} \cdots \sqcup_{\mathrm{in}} \mathring{X}_t^{\ell,n_\ell},$$

common-input union concatenates the corresponding output rows. Consequently,

$$\big\|\tilde{\mathsf{D}}_\circledast \mathring{X}_t^{\ell}(\theta)\big\|_{\mathrm{row}} \le M_\ell. \tag{127}$$

If $\ell = L$, the layer induction is complete. Suppose therefore that $\ell < L$.

By Equation (122), the graphs $\mathring{X}_{t-1}^{\ell}$ and $\mathring{X}_t^{\ell}$ represent uniformly bounded values. Their selected derivatives are uniformly bounded by Equation (127), with the time-zero state having selected derivative zero. Together with Equation (123) and the identity-parameter graph bounds, the source graphs

$$\mathring{X}_{t-1}^{\ell}, \quad \mathring{U}_t^{\ell}, \quad \mathring{X}_t^{\ell}, \quad \mathring{I}[\mathring{\theta}]$$

therefore have uniformly bounded values and selected derivatives.

The next-layer input graph is

$$\mathring{U}_t^{\ell+1} = \big(\mathring{X}_{t-1}^{\ell}, \mathring{U}_t^{\ell}, \mathring{X}_t^{\ell}, \mathring{I}[\mathring{\theta}]\big) \triangleright g_\ell,$$

where g_ℓ is analytic. Lemma 13 therefore yields constants

$$B_{\ell+1}^{U}, \qquad B_{\ell+1}^{U'} \in \mathbb{R}^{\ge 0}$$

satisfying Equation (123) at layer $\ell + 1$. This closes the induction.

At the final layer, Equation (124) gives a uniform bound M_L on every reset and set source graph occurring in the construction of G. Hence, by Theorem 47,

$$\big\|\tilde{\mathsf{D}}_\circledast G(\theta)\big\|_{\mathrm{row}} \le M_L.$$

Thus, the claimed bound holds with $B_r := M_L$.

The bound is uniform over every graph in $\mathcal{G}_\circledast$. Finally, since all spaces are finite-dimensional, norm equivalence converts the row-norm bound into a bound in the norm appearing in Definition 186. Therefore D satisfies $\mathcal{G}_\circledast$-$\tilde{\mathsf{D}}_\circledast$–uBIBSD. □

Corollary 22 (Selected uBIBSD for MinMax state-graph selectors)**.** *Let Γ be a graph selector with*

$$\Gamma\big(\check{D}_u^{\dot{x}}(t)\big) \in \mathcal{G}_\circledast(D; t, u, \dot{x}) \qquad \forall u \in U^\omega,\ \forall \dot{x} \in X,\ \forall t \in \mathbb{T}.$$

Then, the parametric MinMax RNC dynamics D satisfy $\tilde{\mathsf{D}}_\circledast^{\Gamma}$–uBIBSD. In particular, this applies to graph selectors over the subclass of $\mathcal{G}_\circledast$ where recurrent-unit evaluations use sequential or parallel-scan of MinMax-algebraic recurrence graphs—including mixtures thereof.

Proof. Theorem 48 states that D satisfies $\mathcal{G}_\circledast$-$\tilde{\mathsf{D}}_\circledast$–uBIBSD Therefore, Proposition 29 gives $\tilde{\mathsf{D}}_\circledast^{\Gamma}$-uBIBSD, with a constant independent of the selector.

Proposition 50 shows that sequential recurrence graphs belong to the corresponding MinMax-algebraic recurrence-graph classes, while Proposition 52 gives the same conclusion for parallel-scan recurrence graphs. Hence selectors based on either construction, or on admissible mixtures of them, satisfy the hypothesis. □

Remark 35 (Topology independence). Theorem 48 is stronger than a comparison between particular sequential and parallel implementations. For fixed bounded input, initial-state, and parameter sets, the same constant applies to every graph in

$$\mathcal{G}_{\circledast}(D; t, u, \dot{x}) ,$$

uniformly over the time horizon. Thus the selected derivative may depend on the recurrence-evaluation graph at non-differentiable points, but its boundedness does not. *End of Remark 35.*

M.12 BIBLG: Bounded Loss Gradients of MinMax RNCs

We now propagate the graph-uniform state-derivative bound through the analytic analytic output map and the admissible loss. The value bound required in this propagation is supplied by uBIBS from Corollary 20; the derivative bound is supplied by Theorem 48.

Context 2. *For this section, we fix an arbitrary parametric MinMax RNC*

$$S \in \mathrm{RNC}\left(\mathcal{U}^{\mathrm{P}}_{\mathrm{AN}}[\mathrm{MinMax}]\right), \quad S = \langle X, U, f, \dot{x}, Y, h; \Theta^f, \Theta^h \rangle \ \textit{with} \ (\Theta^f \times \Theta^h) =: \Theta \subseteq \mathbb{R}^{d_f} \times \mathbb{R}^{d_h}.$$

Remark 36 (Analytic functions of MinMax RNCs). We recall that the output function h is analytic by the definition of MinMax RNCs, and the same is true for all functions of S beyond the Min and Max operations used in its recurrence—e.g., the set and reset functions of each unit are analytic. *End of Remark 36.*

Definition 239 (MinMax-algebraic objective computation graphs)**.** We define MinMax-algebraic objective computation graphs and then the resulting class.

SubDefinition 239.1. Say we are given:

- an admissible loss function $\ell \in \mathfrak{L}^S$,[13] having analytic ambient extension $\ell^{\uparrow}$;
- a sample $z = (u, y) \in \mathcal{Z}^S$ for S,[14] with $u = (u_t)_{t=1}^T$ for $T \in \mathbb{T}$;
- the infinite extension $uu_T^{\omega} \in U^{\omega}$ of the finite input sequence u given by

$$uu_T^{\omega} \ := \ uu_T, u_T, \ldots, u_T, \ldots \ = \ u_1, u_2, \ldots, u_T, u_T, \ldots, u_T, \ldots$$

  namely, it is the input sequence obtained from u by repeating the last input element u_T indefinitely after step T.

Let $\mathring{\theta}^f$ and $\mathring{\theta}^h$ be node-vectors of the form

$$\mathring{\theta}^f = \left(\mathring{\theta}^f_1, \ldots, \mathring{\theta}^f_{d_f}\right), \qquad \mathring{\theta}^h = \left(\mathring{\theta}^h_1, \ldots, \mathring{\theta}^h_{d_h}\right).$$

All state graphs below have $\mathring{\theta} := \left(\mathring{\theta}^f, \mathring{\theta}^h\right)$ as their input node-vector.

The *MinMax-algebraic objective computation-graph class* for given S, z and ℓ is denoted by

$$\mathcal{G}^{\mathrm{OBJ}}_{\circledast}(S; \ell, z)$$

and it consists of every computation graph G for which there exist computation graphs

$$\mathring{X}^-, \ \mathring{X}^+, \ \mathring{U}_T, \ \mathring{P}, \ \mathring{H}, \ \mathring{Y},$$

[13] See Definition 189.
[14] See Definition 189.

called *constituent graphs* of G, such that:[15,16]

$$\begin{aligned}
\mathring{X}^{-} &= G_{\mathrm{c}}^{\mathrm{p}}[\dot{x}, \mathring{\theta}^{f}] && \text{when } T = 1;\\
\mathring{X}^{-} &\in \mathcal{G}_{\circledast}(D; T-1, uu_T^{\omega}, \dot{x}) && \text{when } T \geq 2;\\
\mathring{X}^{+} &\in \mathcal{G}_{\circledast}(D; T, uu_T^{\omega}, \dot{x})\,;\\
\mathring{U}_T &= G_{\mathrm{c}}^{\mathrm{p}}[u_T, \mathring{\theta}^{f}];\\
\mathring{P} &= \mathring{X}^{-} \sqcup_{\mathrm{in}} \mathring{U}_T \sqcup_{\mathrm{in}} \mathring{X}^{+};\\
\mathring{H} &= \mathring{P} \triangleright h;\\
\mathring{Y} &= G_{\mathrm{c}}^{\mathrm{p}}[y, \mathring{\theta}];\\
G &= (\mathring{H}, \mathring{Y}) \triangleright \ell^{\uparrow}.
\end{aligned}$$

SubDefinition 239.2 (Joint graph class). The *MinMax-algebraic objective computation-graph class* for the given MinMax RNC S is

$$\mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S) := \big(\mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S; \ell, z)\big)_{\ell \in \mathfrak{L}^S,\, z \in \mathcal{Z}^S}\,.$$

SubDefinition 239.3 (Indexed graph family). The *MinMax-algebraic objective indexed computation-graph family* for the given MinMax RNC S is

$$\mathcal{S}_{\circledast}\big[\mathcal{L}_{S,z}^{\ell}\big] := \mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S; \ell, z) \qquad \forall \ell \in \mathfrak{L}^S,\ \forall z \in \mathcal{Z}^S\,.$$

End of Definition 239.

Remark 37 (Graph choices in objective-graph constructions). Objective-graph constructions inherit the possible non-uniqueness of the state graphs from which they are built. In particular, by Remark 28, membership of a state–computation-graph class $\mathcal{G}_{D;T,u,\dot{x}}$ in the general class $\mathcal{G}_{\circledast}(D; T, u, \dot{x})$ does not in general determine a unique choice of the constituent recurrence subgraphs of a graph

$$G \in \mathcal{G}_{D;T,u,\dot{x}} \in \mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S; \ell, z)\,;$$

therefore, such graph G need not be uniquely determined by its class $\mathcal{G}_{D;T,u,\dot{x}}$.

The above does not affect the correctness result below (Theorem 49). In fact, every admissible construction uses state graphs representing the same corresponding RNC states, and hence represents the same objective function. When the state computations are restricted to the sequential or parallel-scan constructions, the corresponding recurrence graphs are determined once their source graphs are fixed—as described in Remark 30 and Remark 32. *End of Remark 37.*

Remark 38 (Irrelevance of the specific infinite extension of the input sequence). The specific extension uu_T^{ω} of the finite sequence $u = u_1, \ldots, u_T$ is immaterial, since the state computation-graph class at horizon $t \in [1..T]$ depends on the input sequence only through the t-step prefix $u_1, \ldots, u_t$ of u, which is also the t-step prefix of the extension uu_T^{ω}. *End of Remark 38.*

Theorem 49 (Correctness of MinMax-algebraic objective graphs)**.** *Let $\ell \mathfrak{L}^S$, let $\ell^{\uparrow}$ be an analytic ambient extension of ℓ, let $z = (u, y) \in \mathcal{Z}^S$, and let $uu_T^{\omega} \in U^{\omega}$ be the infinite extension of u.*[17] *Then, every graph $G \in \mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S; \ell, z)$ is an AMM-graph representing $\mathcal{L}_{S,z}$. Consequently,*

$$\mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S; \ell, z) \subseteq \mathcal{S}_{\circledast}\big[\mathcal{L}_{S,z}^{\ell}\big]\,.$$

Proof. Let $G \in \mathcal{G}_{\circledast}^{\mathrm{OBJ}}(S; \ell, z)$ and fix arbitrary constituent graphs satisfying Definition 239.

Let us introduce the parameter-to-state function for S and on the fixed input sequence uu^{ω}, given by

$$\chi : \Theta^f \times \mathbb{T} \to X\,, \qquad \chi(\theta_f; t) := \check{S}[uu^{\omega}, t](\theta_f)\,.$$

[15] Analytic extension of computation graphs are introduced in Definition 169.
[16] Canonical constant computation graphs are introduced in Definition 155.
[17] Same context as Definition 239.

By Proposition 48, $\mathring{X}^+$ is an AMM-graph representing

$$\theta^f \longmapsto \chi(\theta^f; T).$$

When $T \geq 2$, the same proposition shows that $\mathring{X}^-$ is an AMM-graph representing

$$\theta^f \longmapsto \chi(\theta^f; T-1).$$

When $T = 1$, $\mathring{X}^-$ is the canonical constant graph for $\dot{x}$ and is therefore analytic.

Similarly, $\mathring{U}_T$ is a constant analytic graph. Hence their common-input union $\mathring{P}$ is an AMM-graph representing

$$\theta^f \longmapsto \big(\chi(\theta^f; T-1), u_T, \chi(\theta^f; T)\big).$$

Since h is analytic, the analytic extension

$$\mathring{H} = (\mathring{P}) \triangleright h$$

is an AMM-graph. By correctness of analytic graph extension (Proposition 26),

$$\hat{H}(\theta^f, \theta^h) \;=\; h\Big(\big(\chi(\theta^f; T-1),\, u_T,\, \chi(\theta^f; T)\big);\, \theta^h\Big) \;=\; \check{S}(u; \theta^f, \theta^h)\,.$$

The target graph $\mathring{Y}$ is constant and hence analytic. Since $\ell^\uparrow$ is analytic,

$$G = (\mathring{H}, \mathring{Y}) \triangleright \ell^\uparrow$$

is an AMM-graph and

$$\hat{G}(\theta^f, \theta^h) \;=\; \ell^\uparrow\left(\hat{H}(\theta^f, \theta^h), y\right) \;=\; \ell\Big(\check{S}(u; (\theta^f, \theta^h)),\, y\Big) \;=\; \mathcal{L}_{S,z}(\theta^f, \theta^h)\,.$$

Thus $G \in \mathcal{G}_{\circledast}[\mathcal{L}_{S,z}]$, as required. □

The following is a more precise restatement of Theorem 8.

Theorem 50 (Uniform bounded loss derivatives of MinMax RNCs)**.** *Let us consider an admissible loss function $\ell \in \mathfrak{L}^S$ and a constant $r \in \mathbb{R}^{\geq 0}$.*

Then, there exists a constant $B_{\ell,r} \in \mathbb{R}^{\geq 0}$ such that the following holds:

- *for every sample $z = (u, y) \in \mathcal{Z}^S_{\leq r}$, the following claims hold:*

  (1) for every graph $G \in \mathcal{G}^{\text{OBJ}}_{\circledast}(S; \ell, z)$,

  $$\sup_{\theta \in \Theta_{\leq r}} \big\|\tilde{\mathsf{D}}_{\circledast} G(\theta)\big\| \leq B_{\ell,r}\,; \tag{128}$$

  (2) for every graph $G \in \mathcal{G}^{\text{OBJ}}_{\circledast}(S; \ell, z)$, the function $\tilde{\mathsf{D}}_{\circledast} G$ is an intensional-derivative function of the objective function $\mathcal{L}^\ell_{S,z}$; namely,

  $$\tilde{\mathsf{D}}_{\circledast} \mathcal{L}^\ell_{S,z} \in \partial_\bullet \mathcal{L}^\ell_{S,z}\,;$$

  (3) there exists a full-measure subset $\hat{\Theta}_{r,z} \subseteq \Theta_{\leq r}$ such that

  $$\mathcal{L}^\ell_{S,z} \text{ is differentiable on } \hat{\Theta}_{r,z} \quad \text{and} \quad \sup_{\theta \in \hat{\Theta}_{r,z}} \big\|\mathsf{D}\mathcal{L}^\ell_{S,z}(\theta)\big\| \leq B_{\ell,r}\,; \tag{129}$$

- *consequently, for every graph selector Γ satisfying*

  $$\Gamma\big(\mathcal{L}^\ell_{S,z}\big) \in \mathcal{G}^{\text{OBJ}}_{\circledast}(S; \ell, z) \qquad \forall z \in \mathcal{Z}^S\,,$$

  and for every sample $z \in \mathcal{Z}^S$, the function $\tilde{\mathsf{D}}^\Gamma_{\circledast} \mathcal{L}^\ell_{S,z}$ is a selected intensional derivative of $\mathcal{L}^\ell_{S,z}$, namely:

  $$\tilde{\mathsf{D}}^\Gamma_{\circledast} \mathcal{L}^\ell_{S,z} \in \partial_\bullet \mathcal{L}^\ell_{S,z}\,;$$

• *furthermore,*

$$\sup_{\theta\in\Theta_{\leq r}} \left\|\tilde{\mathsf{D}}^{\Gamma}_{\circledast}\mathcal{L}^{\ell}_{S,z}(\theta)\right\| \leq B_{\ell,r} \qquad \forall z \in \mathcal{Z}^{S}_{\leq r}\,. \tag{130}$$

Remark 39 (Uniformity of the bound). The bounding constant $B_{\ell,r}$ only depends on the considered MinMax RNC S, loss function ℓ, and constant r bounding samples and parameters. In particular, the bound holds uniformly for every admissible graph-selector Γ and every sequence length T.

Proof of Theorem 50. We can write S as an analytic extension, as follows:

$$S = (D, \dot{x}) \triangleright h, \qquad \Theta = \Theta^{f} \times \Theta^{h}.$$

Fix an admissible ℓ and $r \in \mathbb{R}^{\geq 0}$, and let

$$z = (u, y) \in \mathcal{Z}^{S}_{\leq r}, \qquad u = u_1 \cdots u_T.$$

Set

$$\rho_r := \max\left\{r, \|\dot{x}\|\right\}.$$

The continuation uu^{ω}_{T} of Definition 239 satisfies

$$uu^{\omega}_{T} \in U^{\omega}_{\leq \rho_r}.$$

We first prove Point (1). By Corollary 20 and Lemma 14, the output-context parameter functions are uniformly bounded on $\Theta^{f}_{\leq r}$, uniformly over $z \in \mathcal{Z}^{S}_{\leq r}$. Thus there exists $B^{P}_{r} \in \mathbb{R}^{\geq 0}$ such that every context graph $\mathring{P}$ occurring in Definition 239 satisfies

$$\sup_{\theta^{f}\in\Theta^{f}_{\leq r}} \left\|\hat{\mathring{P}}(\theta^{f})\right\| \leq B^{P}_{r}. \tag{131}$$

Furthermore, Theorem 48, applied at radius ρ_r, uniformly bounds the $\tilde{\mathsf{D}}_{\circledast}$-pseudo-derivatives of every admissible choice of $\mathring{X}^{-}$ and $\mathring{X}^{+}$. Since $\mathring{U}_T$ is parameter-independent and common-input union concatenates output rows, there exists $B^{P'}_{r} \in \mathbb{R}^{\geq 0}$ such that every admissible context graph $\mathring{P}$ satisfies

$$\sup_{\theta^{f}\in\Theta^{f}_{\leq r}} \left\|\tilde{\mathsf{D}}_{\circledast}\mathring{P}(\theta^{f})\right\| \leq B^{P'}_{r}. \tag{132}$$

Here and below, finite-dimensional norm equivalence is used whenever passing from the row norm of Theorem 48 to the operator norm.

The Θ^{h}-parameter input of the output graph is the canonical coordinate projection from $\Theta = \Theta^{f} \times \Theta^{h}$. Hence its values and derivatives are uniformly bounded on $\Theta_{\leq r}$, as in Corollary 18. Therefore, applying Lemma 13 to the analytic output map h, with the index set ranging over all samples and all admissible constituent graph choices, yields constants

$$B^{H}_{r}, \qquad B^{H'}_{r} \in \mathbb{R}^{\geq 0}$$

such that every admissible output graph $\mathring{H}$ satisfies

$$\sup_{\theta\in\Theta_{\leq r}} \left\|\hat{\mathring{H}}(\theta)\right\| \leq B^{H}_{r}, \tag{133}$$

$$\sup_{\theta\in\Theta_{\leq r}} \left\|\tilde{\mathsf{D}}_{\circledast}\mathring{H}(\theta)\right\| \leq B^{H'}_{r}. \tag{134}$$

The target graph $\mathring{Y}$ satisfies

$$\|y\| \leq r, \qquad \tilde{\mathsf{D}}_{\circledast}\mathring{Y}(\theta) = 0.$$

Since $\ell^{\uparrow}$ is the designated analytic ambient extension of ℓ, a second application of Lemma 13, now to the analytic extension of $(\mathring{H}, \mathring{Y})$ by $\ell^{\uparrow}$, yields $B_r \in \mathbb{R}^{\geq 0}$ such that every

$$G \in \mathcal{G}^{\mathrm{OBJ}}_{\circledast}(S; \ell, z)$$

satisfies

$$\sup_{\theta\in\Theta_{\le r}} \left\|\tilde{\mathsf{D}}_{\circledast} G(\theta)\right\| \le B_r.$$

The constant B_r is uniform over z and over all admissible objective-graph and constituent-graph choices. This proves Point (1).

We next prove Point (2). Let

$$G \in \mathcal{G}_{\circledast}^{\text{OBJ}}(S;\ell,z)\,.$$

By Theorem 49,

$$G \in \mathcal{G}_{\circledast}\big[\mathcal{L}_{S,z}^{\ell}\big]\,,$$

and in particular $\mathcal{L}_{S,z}^{\ell} \in \mathcal{F}_{\circledast}$. Consider the singleton class

$$\mathcal{Q}_z := \{\mathcal{L}_{S,z}^{\ell}\},$$

and the graph selector Γ_G defined by

$$\Gamma_G[\mathcal{L}_{S,z}^{\ell}] := G.$$

By Theorem 45,

$$\tilde{\mathsf{D}}_{\circledast} G = \tilde{\mathsf{D}}_{\circledast}^{\Gamma_G} \mathcal{L}_{S,z}^{\ell}$$

is an intensional derivative of $\mathcal{L}_{S,z}^{\ell}$. This proves Point (2).

We finally prove Point (3). Fix any

$$G \in \mathcal{G}_{\circledast}^{\text{OBJ}}(S;\ell,z)\,.$$

By Point (2),

$$\tilde{\mathsf{D}}_{\circledast} G \in \partial_{\bullet}\mathcal{L}_{S,z}^{\ell}.$$

Hence, by Proposition 3, there exists a full-measure subset

$$\hat{\Theta}_z \subseteq \Theta$$

such that

$$\mathcal{L}_{S,z}^{\ell} \text{ is differentiable at every } \theta \in \hat{\Theta}_z, \qquad \mathsf{D}\mathcal{L}_{S,z}^{\ell}(\theta) = \tilde{\mathsf{D}}_{\circledast} G(\theta) \quad \forall \theta \in \hat{\Theta}_z.$$

Let

$$\hat{\Theta}_{r,z} := \hat{\Theta}_z \cap \Theta_{\le r}.$$

By Fact 20, $\hat{\Theta}_{r,z}$ is a full-measure subset of $\Theta_{\le r}$. Furthermore, Point (1) gives

$$\mathcal{L}_{S,z}^{\ell} \text{ is differentiable at every } \theta \in \hat{\Theta}_{r,z} \qquad \text{and} \qquad \sup_{\theta\in\hat{\Theta}_{r,z}} \left\|\mathsf{D}\mathcal{L}_{S,z}^{\ell}(\theta)\right\| \le B_r. \tag{135}$$

This proves Point (3).

Finally, let Γ satisfy

$$\Gamma\big(\mathcal{L}_{S,z}^{\ell}\big) \in \mathcal{G}_{\circledast}^{\text{OBJ}}(S;\ell,z) \qquad \forall z \in \mathcal{Z}^S.$$

For every $z \in \mathcal{Z}^S$, Point (2) applied to

$$G = \Gamma\big(\mathcal{L}_{S,z}^{\ell}\big)$$

shows that

$$\tilde{\mathsf{D}}_{\circledast}^{\Gamma} \mathcal{L}_{S,z}^{\ell}$$

is an intensional derivative of $\mathcal{L}_{S,z}^{\ell}$. Furthermore, Point (1) gives

$$\sup_{\theta\in\Theta_{\le r}} \left\|\tilde{\mathsf{D}}_{\circledast}^{\Gamma} \mathcal{L}_{S,z}^{\ell}(\theta)\right\| \le B_r \qquad \forall z \in \mathcal{Z}_{\le r}^S. \tag{136}$$

The constant B_r is independent of Γ because Point (1) holds uniformly over the complete objective computation-graph class. □

Theorem 51 (BIBLG properties of MinMax RNCs)**.** *For every loss function $\ell \in \mathfrak{L}^S$, the following holds:*

- *S satisfies $\mathcal{G}^{\mathrm{OBJ}}_{\circledast}(S)$ -$\tilde{\mathsf{D}}_{\circledast}$-BIBLG;*
- *for every graph selector Γ satisfying*

$$\Gamma\big(\mathcal{L}^{\ell}_{S,z}\big) \in \mathcal{G}^{\mathrm{OBJ}}_{\circledast}(S;\ell,z) \qquad \forall z \in \mathcal{Z}^S,$$

we have that S satisfies $\tilde{\mathsf{D}}^{\Gamma}_{\circledast}$-BIBLG, with constants independent of Γ;

- *S satisfies ae-BIBLG.*

Proof. By Theorem 49, every graph in $\mathcal{G}^{\mathrm{OBJ}}_{\circledast}(S)$ represents the corresponding objective function and is an AMM-graph. Furthermore, Points (1) and (2) of Theorem 50 give, respectively, the graph-uniform derivative bound and the intensional-derivative property required by Definition 195. Hence S satisfies $\mathcal{G}^{\mathrm{OBJ}}_{\circledast}(S)$ -$\tilde{\mathsf{D}}_{\circledast}$-BIBLG.

The second claim follows from Proposition 31, which also shows that the corresponding constants can be chosen independently of Γ.

Finally, Point (3) of Theorem 50 is precisely the full-measure differentiability and uniform derivative bound required by Definition 192. Hence S satisfies ae-BIBLG. □

Remark 40 (Relation to Theorem 8). Theorem 50 gives the graph-uniform formulation underlying Theorem 8. In particular, the derivative bound is established uniformly over the complete objective computation-graph family before any graph selector is chosen. Consequently, every admissible graph selector inherits the same bounding constant. *End of Remark 40.*

M.13 Non-necessarily Vanishing State-to-State Derivatives

We prove Theorem 9 uniformly over MinMax-algebraic *recurrence-evaluation graphs*. In this subsection, reset and set coefficients are fixed once an input sequence has been chosen; the initial state is the variable with respect to which derivatives are taken. The admissible graphs therefore contain the initial-state coordinates and these coefficients as their only scalar sources, with only MinMax reductions introduced during recurrence evaluation. This includes sequential, composed-pair, and parallel-scan evaluations, and their MinMax-algebraic rearrangements. It does not mean all analytic-MinMax graphs representing the same state-to-state function.

Definition 240 (Initial-state-variable MinMax recurrence graphs)**.** Let $D = \langle X, U, f \mid R, s\rangle$ be an n-dimensional MinMax recurrent unit, let $t \in \mathbb{N}^{>0}$, and fix $u_1, \dots, u_t \in U$. Write $u := (u_1, \dots, u_t)$ and let $q_t : X \to X$ be the corresponding state-to-state map. Let $\bar{q}_t : \mathbb{R}^n \to \mathbb{R}^n$ be its canonical ambient extension, obtained by evaluating the same MinMax recurrence formula on arbitrary initial vectors in $\mathbb{R}^n$. Thus $\bar{q}_t|_X = q_t$, and derivatives of q_t below are understood through this ambient extension. Let $C_{u,t}$ be the set of scalar entries of $R(u_k)$ and $s(u_k)$ for $1 \le k \le t$. Let Q^{id} be the identity computation graph on the initial-state variable $x \in \mathbb{R}^n$. For each k, let Q^R_k and Q^s_k be canonical constant computation graphs representing $R(u_k)$ and $s(u_k)$ on the same input space. The *initial-state-variable MinMax recurrence-graph family* for the ambient extension of q_t is

$$\mathcal{G}^{x}_{D,u,t} := \big\{ G \in \mathcal{E}_{\circledast}\big(Q^{\mathrm{id}}, Q^R_1, \dots, Q^R_t, Q^s_1, \dots, Q^s_t\big) \;\big|\; \hat{G} = \bar{q}_t \big\}.$$

The source-output nodes of such a graph are the initial-state coordinates and the fixed recurrence coefficients; its new nodes are min/max reductions. A graph selector is called *recurrence-selected* if, at every horizon t, it selects a graph from $\mathcal{G}^{x}_{D,u,t}$.

Lemma 32 (Graph-independent routing of initial-state derivatives)**.** *Fix D, u, t as above and $x_0 \in X$. Suppose that the coordinates $(x_0)_1, \dots, (x_0)_n$ are pairwise distinct, that none belongs to $C_{u,t}$, and that each coordinate of $q_t(x_0)$ is one of the coordinates of x_0. Then there exists a unique map $\pi_t : \{1, \dots, n\} \to \{1, \dots, n\}$ such that*

$$(q_t(x_0))_i = (x_0)_{\pi_t(i)} \qquad \forall i \in \{1, \dots, n\}. \tag{137}$$

The map q_t is ordinarily differentiable at x_0, and, for every recurrence-selected MinMax graph selector Γ,

$$\tilde{\mathsf{D}}^{\Gamma}_{\circledast} q_t(x_0) = \mathsf{D} q_t(x_0) = P_{\pi_t}, \qquad (P_{\pi_t})_{ij} := \mathbf{1}[j = \pi_t(i)]. \tag{138}$$

If π_t is a permutation, then $\|P_{\pi_t}\| = 1$ in the Euclidean operator norm.

Proof. Fix a graph $G \in \mathcal{G}^{x}_{D,u,t}$. Every scalar source of G is either a coordinate function $x \mapsto x_j$, with derivative $e_j^\top$, or a constant function with value in $C_{u,t}$, with derivative zero. Every other scalar node applies a finite minimum or maximum to predecessor values. Hence, at x_0, the value of every scalar node belongs to

$$C_{u,t} \cup \{(x_0)_1, \dots, (x_0)_n\}. \tag{139}$$

Fix an output coordinate i. By assumption its value is $(x_0)_{\pi_t(i)}$, and the index $\pi_t(i)$ is unique. At any min/max node, the MinMax pseudo-derivative puts nonnegative weights summing to one on the active predecessors, i.e., those whose values equal the node's value. Following an active predecessor therefore preserves the scalar value. Starting at output coordinate i and following active predecessors to the sources, every reached source must consequently have value $(x_0)_{\pi_t(i)}$. It cannot be a coefficient source, since $(x_0)_{\pi_t(i)} \notin C_{u,t}$; nor can it be another initial-state coordinate, since the coordinates of x_0 are pairwise distinct. All active source-to-output paths therefore have derivative $e_{\pi_t(i)}^\top$. By the chain rule for the selected MinMax pseudo-derivative, taking convex combinations of these identical rows yields

$$\big(\tilde{\mathsf{D}}_{\circledast} G(x_0)\big)_{i,:} = e_{\pi_t(i)}^\top.$$

This argument allows ties between distinct computational paths as long as they carry the same initial-state coordinate. Since G was arbitrary, the selected Jacobian is P_{π_t} for every recurrence-selected graph.

We also establish ordinary differentiability. Because the graph is finite, it has only finitely many scalar sources, each either constant or a coordinate projection. At x_0, all different initial-state coordinates are separated from one another and from every coefficient value. Thus there is an open neighbourhood V of x_0 on which the order of any two source functions that have different values at x_0 does not change. Source functions agreeing at x_0 are either identical coordinate projections or constant functions; if they are identical projections, their values also agree throughout V. Consequently, by induction through the finite min/max graph, each output coordinate agrees on V with the same initial-state coordinate projection selected at x_0. Hence

$$q_t(x) = P_{\pi_t} x \qquad \forall x \in V,$$

proving $\mathsf{D} q_t(x_0) = P_{\pi_t}$. If π_t is a permutation, then P_{π_t} is orthogonal, so $\|P_{\pi_t}\| = 1$. □

M.13.1 Proof of Theorem 9

Theorem 9 (Non-vanishing state-to-state derivatives)**.** *For every $\epsilon \in \mathbb{R}^{>0}$, there exists $(\mathcal{D}_n)_{n\geq 1}$ satisfying the following jointly: (i) each $D \in \mathcal{D}_n$ is a MinMax Neuron whose state space contains an n-dimension hypercube $\tilde{X}_D$ of side length ϵ; (ii) every identity-reset semiautomaton can be realised by a Neuron of $\mathcal{D}_1$; (iii) every n-degree permutation semiautomaton can be realised by a Neuron of $\mathcal{D}_n$; (iv) each $D \in \mathcal{D}_n$ admits an infinite input squence u such that, for every selector Γ over $\mathcal{G}_{\circledast}$,*

$$\tilde{\mathsf{D}}^{\Gamma}_{\circledast} \check{D}^u_t(x_0) = \mathsf{D}\check{D}^u_t(x_0) \quad \textit{and} \quad \|\tilde{\mathsf{D}}^{\Gamma}_{\circledast} \check{D}^u_t(x_0)\| = 1 \qquad \forall x_0 \in \tilde{X}_D \quad \forall t \in \mathbb{N}^{>0}.$$

Proof. Fix $\epsilon \in \mathbb{R}^{>0}$. We construct the classes $(\mathcal{D}_n)_{n\geq 1}$ using the identity-reset and permutation realisations of Theorems 3 and 4, choosing their state regions and coefficient margins as specified below. In particular, the choices are made so that every neuron in the classes has a hypercube of side length ϵ and a fixed infinite input witnessing the asserted derivative identities.

Scalar identity-reset realisations. Let $A = (Q, \Sigma, \delta)$ be an arbitrary identity-reset semiautomaton with $Q = \{q_1, \dots, q_m\}$. Choose pairwise-disjoint closed intervals

$$I_j = [c_j - \epsilon/2, c_j + \epsilon/2] \qquad (1 \leq j \leq m),$$

with positive gaps between them, and set $X := \bigcup_{j=1}^m I_j$. Choose $L, H \in \mathbb{R}$ with

$$L < \min X < \max X < H. \tag{140}$$

Assign distinct input components to the symbols of Σ. If necessary, adjoin one extra input component, represented by an input $u_\star$, whose transformation is the identity; this extra component need not

represent a symbol of A. On an identity component prescribe the scalar coefficients $(R, s) = (H, L)$; on a component resetting to q_j prescribe $(R, s) = (L, c_j)$. The resulting MinMax unit has

$$f(x,u) = \max\{\min\{R(u), x\}, s(u)\} \\ = \begin{cases} x, & \text{on an identity component,} \\ c_j, & \text{on a component resetting to } q_j, \end{cases} \qquad x \in X.$$

Thus its separated state components I_j realise A, and $u_\star$ acts as the identity if it was adjoined.

In passing to a MinMax Neuron, choose MLP coefficient functions approximating the prescribed functions uniformly on the finitely many compact input components, with error small compared with the strict gaps in (140) and with the distance from c_j to the boundary of I_j. The realised neuron D then still satisfies, on each identity component,

$$R(u) > \max X, \qquad s(u) < \min X, \tag{141}$$

and, on a component resetting to q_j,

$$R(u) < \min X, \qquad s(u) \in \operatorname{int}(I_j).$$

Hence the identity transitions remain *exactly* identity, and each reset still reaches its prescribed state component. Such neurons realise A as required by Theorem 3.

Include all chosen scalar neurons in $\mathcal{D}_1$, retaining one identity input $u_\star$ for each. For each such D, choose

$$\tilde{X}_D := I_1, \qquad u = (u_\star, u_\star, \ldots).$$

Equation (141) implies $f(x, u_\star) = x$ on the open interval $(s(u_\star), R(u_\star))$, which contains X. Therefore, for every $t \in \mathbb{N}^{>0}$,

$$q_t(x) = x \qquad \text{on a neighbourhood of } \tilde{X}_D. \tag{142}$$

In particular, $\mathsf{D}q_t(x_0) = [1]$ for all $x_0 \in \tilde{X}_D$. All recurrence-coefficient values along this input are strictly outside X and hence outside $\tilde{X}_D$. Lemma 32 (with $n = 1$) shows that, for every recurrence-selected graph selector,

$$\tilde{\mathsf{D}}^{\Gamma}_{\circledast} q_t(x_0) = \mathsf{D}q_t(x_0) = [1], \qquad \left\| \tilde{\mathsf{D}}^{\Gamma}_{\circledast} q_t(x_0) \right\| = 1. \tag{143}$$

This verifies the required assertion for every $D \in \mathcal{D}_1$. In addition, degree-one permutation semiautomata are identity semiautomata and are covered by these scalar realisations.

Permutation realisations in dimensions $n \geq 2$. Fix $n \geq 2$ and let A be an arbitrary permutation semiautomaton of degree n. As in Theorem 4, we may represent its input transformations by coordinate permutations π_σ on $\{1, \ldots, n\}$. Choose pairwise-disjoint closed intervals

$$I_j = [c_j - \epsilon/2, c_j + \epsilon/2] \qquad (1 \leq j \leq n),$$

with positive gaps, put $K := \bigcup_{j=1}^{n} I_j$ and $X := K^n$, and choose

$$L < \min K < \max K < H. \tag{144}$$

For a symbol $\sigma \in \Sigma$ and its corresponding input component, prescribe

$$R(u)_{ij} := \begin{cases} H, & j = \pi_\sigma(i), \\ L, & j \neq \pi_\sigma(i), \end{cases} \qquad s(u)_i := L. \tag{145}$$

Then for all $x \in X$,

$$f(x,u)_i = \max\left\{ \max_{1 \leq j \leq n} \min\{R(u)_{ij}, x_j\}, s(u)_i \right\} = x_{\pi_\sigma(i)}. \tag{146}$$

This is exactly the coordinate-permutation realisation used in Theorem 4.

As in the scalar case, choose MLP coefficient functions approximating the prescribed coefficients uniformly, with sufficiently small error that every coefficient formerly equal to H remains strictly greater than $\max K$, and every coefficient formerly equal to L remains strictly smaller than $\min K$.

Equation (146) then holds *exactly* for the resulting MinMax Neuron, not merely approximately. Let $\mathcal{D}_n$ consist of these chosen neurons, one for each degree-n permutation semiautomaton, and choose for each one

$$\tilde{X}_D := I_1 \times \cdots \times I_n \subseteq X. \tag{147}$$

This is an n-dimensional hypercube of side length ϵ; every point of it has pairwise-distinct coordinates.

Choose any infinite input sequence $u = (u_1, u_2, \ldots)$ using the input components of the constructed neuron. For each k, let P_k denote the permutation matrix of the k-th transition. The equality in (146) holds on a neighbourhood of X (by the strict guard margins), and X is invariant under coordinate permutations. Consequently, for every $t \in \mathbb{N}^{>0}$,

$$q_t(x) = P_t P_{t-1} \cdots P_1 x \qquad \text{on a neighbourhood of } \tilde{X}_D, \tag{148}$$

and hence

$$\mathsf{D}q_t(x_0) = P_t \cdots P_1 \qquad \forall x_0 \in \tilde{X}_D. \tag{149}$$

The product $P_t \cdots P_1$ is a permutation matrix, so its Euclidean operator norm is one.

Finally, every scalar coefficient of $R(u_k)$ and $s(u_k)$ is strictly below $\min K$ or strictly above $\max K$, whereas each coordinate of every $x_0 \in \tilde{X}_D$ belongs to K. The coordinates of x_0 are pairwise distinct, and (148) shows that each coordinate of $q_t(x_0)$ is one of those coordinates. Lemma 32 therefore applies to every horizon t and every recurrence-selected selector, giving

$$\tilde{\mathsf{D}}^{\Gamma}_{\circledast} q_t(x_0) = \mathsf{D}q_t(x_0) = P_t \cdots P_1, \tag{150}$$

$$\left\| \tilde{\mathsf{D}}^{\Gamma}_{\circledast} q_t(x_0) \right\| = 1 \qquad \forall x_0 \in \tilde{X}_D. \tag{151}$$

Conclusion. Every member of $\mathcal{D}_n$ is a MinMax Neuron with an n-dimensional hypercube of side length ϵ in its state space. By construction, $\mathcal{D}_1$ realises every identity-reset semiautomaton and also covers degree-one permutation semiautomata; for each $n \geq 2$, $\mathcal{D}_n$ realises every degree-n permutation semiautomaton. Equations (143) and (150) establish the stated graph-uniform derivative identity and unit norm for every $t \in \mathbb{N}^{>0}$ and every $x_0 \in \tilde{X}_D$, completing the proof. □

M.14 Deferred Proofs for MinMax Pseudo-Derivatives

Proposition 53. *Let $f : X \to \mathbb{R}$ be the function $f(x) = \bigoplus_{i=1}^{n} x_i$. For every $x \in \mathbb{R}^n$ and every $d \in \mathbb{R}^n$,*

$$\lim_{t \downarrow 0} \frac{f(x+td) - f(x)}{t} = \bigoplus_{i \in I(x)} d_i.$$

Proof. Let

$$d_* = \bigoplus_{i \in I(x)} d_i.$$

We show the two following inequalities separately

$$\liminf_{t \downarrow 0} \frac{f(x+td) - f(x)}{t} \geq d_*\,, \tag{152}$$

$$\liminf_{t \downarrow 0} \frac{f(x+td) - f(x)}{t} \leq d_*\,. \tag{153}$$

We first show (152). For every $i \in I(x)$, we have $x_i = f(x)$ and hence

$$f(x+td) \;=\; \bigoplus_{j \in [1..n]} (x_j + td_j) \;\geq\; x_i + td_i \;=\; f(x) + td_i \qquad \forall t \in \mathbb{R}.$$

Thus,

$$\frac{f(x+td) - f(x)}{t} \geq d_i \qquad \forall i \in I(x),\, \forall t \in \mathbb{R}^{>0},$$

hence

$$\frac{f(x+td) - f(x)}{t} \geq d_* \qquad \forall t \in \mathbb{R}^{>0},$$

and hence (152) follows immediately.

We now prove (153). If $I(x) = [1..n]$, then

$$f(x+td) = \bigoplus_{j=1}^{n} x_j + td_j = \bigoplus_{j=1}^{n} f(x) + td_j = f(x) + t \cdot \bigoplus_{j=1}^{n} d_j = f(x) + t \cdot \bigoplus_{i \in I(x)} d_i = f(x) + td_* \quad \forall t \in \mathbb{R}^{>0},$$

and hence

$$\frac{f(x+td) - f(x)}{t} = \frac{f(x) + td_* - f(x)}{t} = \frac{td_*}{t} = d_* \qquad \forall t \in \mathbb{R}^{>0},$$

which immediately implies (153).

Next, let us consider the complementary case when $I(x) \subset [1..n]$, and hence let

$$J(x) := [1..n] \setminus I(x)$$

Since $J(x)$ is finite and $x_j < f(x)$ for every $j \in J(x)$, we have

$$\delta := \min_{j \in J(x)} (f(x) - x_j) > 0.$$

Let us define

$$C := 1 + \bigoplus_{j \in J(x)} |d_j - d_*| > 0.$$

For every $j \in J(x)$ and every t satisfying $0 < t < \delta/C$, we have

$$\begin{aligned} x_j + td_j &= f(x) - (f(x) - x_j) + td_j \\ &\le f(x) - \delta + td_j \\ &= f(x) + td_* - \delta + t(d_j - d_*) \\ &\le f(x) + td_* - \delta + tC \\ &< f(x) + td_* - \delta + \frac{\delta}{C}C \\ &= f(x) + td_* - \delta + \delta \\ &= f(x) + td_*, \end{aligned}$$

overall,

$$x_j + td_j < f(x) + td_* \qquad \forall t \in (0, \delta/C).$$

On the other hand, by definition of d_*, there exists some $i^\star \in I(x)$ such that $d_{i^\star} = d_*$, and hence

$$x_{i^\star} + td_{i^\star} = f(x) + td_* \qquad \forall t \in \mathbb{R}^{>0}.$$

Thus, for all $t \in (0, \delta/C)$, we have

$$f(x + td) = \bigoplus_{i \in I(x)} x_i + td_i \qquad \forall t \in \mathbb{R}^{>0},$$

hence

$$f(x + td) = \bigoplus_{i \in I(x)} x_i + td_i = \bigoplus_{i \in I(x)} f(x) + td_i = f(x) + t \cdot \bigoplus_{i \in I(x)} d_i = f(x) + td_*,$$

and hence

$$\frac{f(x + td) - f(x)}{t} = \frac{f(x) + td_* - f(x)}{t} = \frac{td_*}{t} = d_*.$$

Then (153) follows immediately. □

Proposition 38. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function $f(x) = \bigoplus_{i=1}^n x_i$, let I be the active-variable index function of f (Definition 225), and let $x \in \mathbb{R}^n$. Then, the subdifferential of f at x is given by*

$$\partial_\circ f(x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \bigwedge_{i=1}^n g_i \ge 0,\ \sum_{i \in I(x)} g_i = 1 \right\}.$$

Furthermore, the unique subgradient $g^\star \in \partial_\circ f(x)$ of minimum (Euclidean) norm is given by

$$g^\star = (g_1^\star, \ldots, g_n^\star) \qquad \text{where } g_i^\star = \begin{cases} 1/|I(x)| & \text{if } i \in I(x), \\ 0 & \text{otherwise}, \end{cases}$$

and its norm is $\|g^\star\| = 1/\sqrt{|I(x)|}$.

Proof. By definition, $g \in \partial_\circ f(x)$ if and only if

$$f(y) \ge f(x) + g^\top (y - x) \qquad \text{for all } y \in \mathbb{R}^n.$$

Equivalently,

$$\bigoplus_{j=1}^n y_j \ge f(x) + g^\top (y - x) \qquad \text{for all } y \in \mathbb{R}^n.$$

Now set $y = x + td$, where $t > 0$ and $d \in \mathbb{R}^n$. Then the subgradient inequality becomes

$$\bigoplus_{j=1}^n (x_j + td_j) \ge f(x) + tg^\top d.$$

Subtracting $f(x)$ and dividing by $t > 0$, we obtain

$$\frac{\bigoplus_{j=1}^{n}(x_j + td_j) - f(x)}{t} \geq g^\top d.$$

By Proposition 53 we have

$$\lim_{t\downarrow 0} \frac{\bigoplus_{j=1}^{n}(x_j + td_j) - f(x)}{t} = \bigoplus_{i\in I(x)} d_i,$$

and hence

$$\lim_{t\downarrow 0} \frac{\bigoplus_{j=1}^{n}(x_j + td_j) - f(x)}{t} \geq \lim_{t\downarrow 0} g^\top d$$

yields

$$\bigoplus_{i\in I(x)} d_i \geq g^\top d \qquad \forall d \in \mathbb{R}^n.$$

Therefore every subgradient $g \in \partial_\circ f(x)$ satisfies

$$g^\top d \leq \bigoplus_{i\in I(x)} d_i \qquad \forall d \in \mathbb{R}^n.$$

We now derive the consequences of this inequality. First, take $d = \mathbf{1}$, where $\mathbf{1} = (1, \ldots, 1)$. Then

$$g^\top \mathbf{1} \leq \bigoplus_{i\in I(x)} 1 = 1.$$

Taking $d = -\mathbf{1}$ gives

$$-g^\top \mathbf{1} \leq \bigoplus_{i\in I(x)} (-1) = -1.$$

Hence

$$g^\top \mathbf{1} \geq 1.$$

Combining the two inequalities, we get

$$\sum_{j=1}^{n} g_j = 1.$$

Next, let $j \notin I(x)$. Choose d such that

$$d_\ell = 0 \qquad \forall \ell \in [1..n] \setminus \{j\},$$

while d_j is arbitrary. Then

$$\bigoplus_{i\in I(x)} d_i = 0.$$

The inequality $g^\top d = g_j d_j \leq 0$ must hold for every value of $d_j \in \mathbb{R}$. In particular, the inequalities $g_j \leq 0$ (where $d_j = 1$) and $g_j \geq 0$ (where $d_j = -1$) must hold, and hence $g_j = 0$. As the above applies to each $j \notin I(x)$, we conclude that

$$g_j = 0 \qquad \forall j \notin I(x).$$

Finally, let $i \in I(x)$. If $|I(x)| \geq 2$, choose d such that

$$\begin{aligned} &d_i = -1, \\ &d_\ell = 0 \qquad \text{for each } \ell \in I(x),\ \ell \neq i, \\ &d_j = 0 \qquad \text{for each } j \notin I(x). \end{aligned}$$

Then

$$\bigoplus_{\ell \in I(x)} d_\ell = 0.$$

The inequality $g^\top d \leq 0$ gives

$$-g_i \leq 0,$$

and therefore

$$g_i \geq 0.$$

If $|I(x)| = 1$, say $I(x) = \{i\}$, then since $g_j = 0$ for all $j \neq i$ and $\sum_{j=1}^n g_j = 1$, we get

$$g_i = 1 \geq 0.$$

Thus every subgradient satisfies

$$\begin{aligned} g_i &\geq 0 \qquad \text{for each } i \in I(x), \\ g_i &= 0 \qquad \text{for each } i \notin I(x), \\ \sum_{i \in I(x)} g_i &= 1. \end{aligned}$$

Conversely, suppose $g \in \mathbb{R}^n$ satisfies

$$\begin{aligned} g_i &\geq 0 \qquad \text{for each } i \in I(x), \\ g_i &= 0 \qquad \text{for each } i \notin I(x), \\ \sum_{i \in I(x)} g_i &= 1. \end{aligned}$$

Then, for any $y \in \mathbb{R}^n$,

$$g^\top (y - x) = \sum_{i \in I(x)} g_i (y_i - x_i).$$

Since $x_i = f(x)$ for every $i \in I(x)$, this becomes

$$g^\top (y - x) = \sum_{i \in I(x)} g_i (y_i - f(x)).$$

Also, for every $i \in I(x)$,

$$y_i \leq \bigoplus_{j=1}^n y_j.$$

Therefore,

$$y_i - f(x) \leq \bigoplus_{j=1}^n y_j - f(x).$$

Using $g_i \geq 0$ and $\sum_{i \in I(x)} g_i = 1$, we obtain

$$\begin{aligned} g^\top (y - x) &\leq \sum_{i \in I(x)} g_i \left(\bigoplus_{j=1}^n y_j - f(x) \right) \\ &= \bigoplus_{j=1}^n y_j - f(x). \end{aligned}$$

Hence

$$f(x) + g^\top (y - x) \leq \bigoplus_{j=1}^{n} y_j.$$

Since $f(y) = \bigoplus_{j=1}^{n} y_j$, this is exactly

$$f(y) \geq f(x) + g^\top (y - x).$$

Thus $g \in \partial_\circ f(x)$.

It remains to find the subgradient of minimum (Euclidean) norm. Since $g_i = 0$ for every $i \notin I(x)$, we need to solve

$$\min \sum_{i \in I(x)} g_i^2$$

subject to

$$\bigwedge_{i=1}^{n} g_i \geq 0,$$
$$\sum_{i \in I(x)} g_i = 1.$$

By the Cauchy–Schwarz inequality,

$$\left(\sum_{i \in I(x)} g_i \right)^2 \leq |I(x)| \sum_{i \in I(x)} g_i^2.$$

Since $\sum_{i \in I(x)} g_i = 1$, we get

$$1 \leq |I(x)| \sum_{i \in I(x)} g_i^2.$$

Therefore,

$$\sum_{i \in I(x)} g_i^2 \geq \frac{1}{|I(x)|}.$$

Equality in Cauchy–Schwarz holds if and only if all the active components g_i, $i \in I(x)$, are equal. Since they sum to 1, equality holds precisely when

$$g_i = \frac{1}{|I(x)|} \qquad \text{for all } i \in I(x).$$

Thus the minimum norm subgradient is

$$g_i^\star = \begin{cases} \dfrac{1}{|I(x)|}, & i \in I(x), \\ 0, & i \notin I(x). \end{cases}$$

Finally,

$$\|g^\star\| = \sqrt{\sum_{i \in I(x)} \frac{1}{|I(x)|^2}} = \sqrt{|I(x)| \frac{1}{|I(x)|^2}} = \sqrt{\frac{1}{|I(x)|}} = \frac{1}{\sqrt{|I(x)|}}.$$

This concludes the proof. □

Proposition 39. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function $f(x) = \bigodot_{i=1}^n x_i$, let I be the active-variable index function of f (Definition 225), and let $x \in \mathbb{R}^n$. Then, the superdifferential of f at x is given by*

$$\partial_\circ^+ f(x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in I(x)} g_i = 1 \right\}.$$

Furthermore, the unique supergradient $g^\star \in \partial_\circ f(x)$ of minimum (Euclidean) norm is given by

$$g^\star = (g_1^\star, \ldots, g_n^\star) \qquad \text{where } \; g_i^\star = \begin{cases} 1/|I(x)| & \text{if } i \in I(x), \\ 0 & \text{otherwise}, \end{cases}$$

and its norm is $\|g^\star\| = 1/\sqrt{|I(x)|}$.

Proof. Let $\bar{f} : \mathbb{R}^n \to \mathbb{R}$ be the function $\bar{f}(x_1, \ldots, x_n) = \bigoplus_{i=1}^n x_i$. Let I_f be the active variable index function of f, and let $\bar{I}$ be the active variable index function of $\bar{f}$. Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. By the definition of superdifferential, we have

$$g \in \partial_\circ^+ f(x) \quad \iff \quad -g \in \partial_\circ(-f)(x) \qquad \forall g \in \mathbb{R}^n. \tag{154}$$

Then,

$$-f(x) = -\bigodot_{i=1}^n x_i = \bigoplus_{i=1}^n -x_i = \bar{f}(-x),$$

and, by Proposition 38,

$$\partial_\circ \bar{f}(-x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in \bar{I}(-x)} g_i = 1 \right\}. \tag{155}$$

We have

$$\bar{I}(-x) = \left\{ i \in [1..n] \mid \bigoplus_{i=1}^n -x_i = -x_i \right\} = \left\{ i \in [1..n] \mid \bigodot_{i=1}^n x_i = x_i \right\} = I(x),$$

and hence, from (155), we obtain

$$\partial_\circ \bar{f}(-x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in I(x)} g_i = 1 \right\}.$$

By the calculus of subdifferentials, we have $\partial_\circ(-\bar{f})(x) = -\partial_\circ \bar{f}(x)$, meaning that

$$g \in \partial_\circ(-\bar{f})(x) \quad \iff \quad -g \in \partial_\circ \bar{f}(x) \qquad \forall g \in \mathbb{R}^n,$$

and hence

$$\partial_\circ(-f)(x) \;=\; \partial_\circ(-\bar{f})(-x) \;=\; -\partial_\circ \bar{f}(-x) \;=\; \left\{ -g \in \mathbb{R}^n \;\middle|\; \bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in I(x)} g_i = 1 \right\}.$$

Thus, by (154), we have

$$\partial_\circ^+ f(x) = \left\{ g \in \mathbb{R}^n \;\middle|\; \bigwedge_{i=1}^n g_i \geq 0,\ \sum_{i \in I(x)} g_i = 1 \right\},$$

which is our first claim. Then, our second claim, on the supergradient of minimum norm, follows by Proposition 38. □

N Piecewise-Affine Structure of MinMax Recurrences

Table of Contents

N.1 Active Variables of MinMax Functions 238

N.2 Piecewise Propagate-Reset Structure of MinMax Recurrences 240

N.3 Composition and Relation to Linear Recurrences 247

This appendix shows that MinMax recurrences have piecewise-affine structure.

We first provide a characterisation of MinMax dynamics functions where they are seen as producing updated states by selecting individual components of the given (i) state vector x and (ii) input vectors $R(u), s(u)$. We refer to any component of $x, R(u), s(u)$ as a *source variable*, and to any source variable that is selected within an update as an *active variable*. Then, the above selections induce a finite partition of the joint state-input space, which we call *active-variable partition* since its regions are characterised by sets of active variables, with such set being constant across each region.

On every active-variable region $Z \in \mathcal{Z}$, the MinMax dynamics admits a state-affine representation

$$f(x, u) \;=\; A_Z \cdot x + b_Z(u),$$

with a highly constrained coefficient-offset pair: every row of A_Z is either zero or a standard basis row, and state propagation and input-dependent reset are mutually exclusive coordinatewise. We refer to this structure as *propagate-reset*. Thus, whenever the recurrence is iterated on a forward-invariant state space, its evolution along a trajectory can be written as a switched time-varying affine recurrence

$$x_t \;=\; A_t \cdot x_{t-1} + b_t \qquad\qquad \forall t \in \mathbb{T},$$

whose active coefficients satisfy the propagate-reset constraints.

We develop several consequences of this representation. First, the propagate-reset structure is closed under composition: each state coordinate at a later time is obtained either from a single earlier state coordinate or from a single input-dependent source value introduced at one intermediate time point. In particular, input-dependent contributions do not accumulate additively. This provides a structural explanation for the bounded propagation properties of MinMax recurrences without requiring strict contraction. At points where the active-variable pattern is locally constant, the active matrices also coincide with the state Jacobians; products of these Jacobians remain uniformly bounded, while active propagation paths may preserve derivatives of magnitude one over arbitrarily long horizons.

The same structure yields regularity properties. For every fixed input, the state map is piecewise affine; if the reset and set functions are themselves piecewise affine, then the joint MinMax dynamics is piecewise affine. Likewise, suitable PAP regularity of the source functions is inherited by the MinMax dynamics. We also distinguish the deterministic active-variable routing used in the piecewise representation from the selected derivatives used at switching boundaries, where several source variables may be simultaneously active.

Finally, we study the geometry of the active-variable partition. For a fixed input, changes of active-variable pattern are controlled by a finite family of coordinate-equality and coordinate-threshold hyperplanes. Consequently, every fixed-input section of an active-variable region is the intersection of the state space with a semilinear set. The possible hyperplane orientations are restricted to the coordinate directions and coordinate-difference directions, whereas their threshold offsets are determined by the fixed-input source values. Already in the scalar case, the resulting active-variable regions can realise arbitrary bounded interval interiors, showing that their sizes and placements are not uniformly constrained by the structural form of the MinMax recurrence.

This section develops a linear-system interpretation of MinMax recurrences that helps relate their stability and gradient properties to those of linear RNNs. The key observation is that MinMax is piecewise state-affine: on every active-variable region, each state coordinate is either copied from one coordinate of the previous state or replaced by a current input-dependent value. Thus, along a state sequence $(x_t)_{t\in\mathbb{T}}$ starting from a given state x_0, MinMax behaves as a switched time-varying affine recurrence

$$x_t \;=\; A_t \cdot x_{t-1} + b_t \qquad\qquad \forall t \in \mathbb{T},$$

but with highly constrained recurrent matrices and mutually exclusive state propagation and input injection. This structure permits unit-gain information propagation while preventing additive accumulation, thereby explaining both state stability and bounded, non-necessarily-vanishing state gradients.

N.1 Active Variables of MinMax Functions

Definition 241 (MinMax primitives and functions)**.** For $d \in \mathbb{N}^{>0}$ and $j \in [1..d]$, let

$$\pi_j^d : \mathbb{R}^d \to \mathbb{R}, \qquad \pi_j^d(x) := (x)_j,$$

be the j-th coordinate projection.

The class of *MinMax primitives* is

$$\mathcal{P}_{\mathrm{MM}} := \left\{ \pi_j^d \;\middle|\; d \in \mathbb{N}^{>0},\ j \in [1..d] \right\} \cup \left\{ \mathsf{min}_d, \mathsf{max}_d \mid d \in \mathbb{N}^{>0} \right\},$$

The class $\mathcal{F}_{\mathrm{MM}}$ of *MinMax functions* is the closure of $\mathcal{P}_{\mathrm{MM}}$. *End of Definition 241.*

Definition 242 (Closure derivation)**.** Let $\mathcal{P}$ be a class of functions.

A *closure derivation* relative to the class $\mathcal{P}$ is defined inductively as follows:

- for $f \in \mathcal{P}$, the tuple $\langle \mathsf{primitive}, f \rangle$, called the *primitive derivation* of f;
- for $f = (f_1, \ldots, f_k) \in \mathrm{cl}(\mathcal{P})$, a tuple $\langle \mathsf{tupling}, \delta_1, \ldots, \delta_k \rangle$ where $\delta_1, \ldots, \delta_k$ are closure derivations of $f_1, \ldots, f_k$, respectively;
- for $f = (h \circ g) \in \mathrm{cl}(\mathcal{P})$, a tuple $\langle \mathsf{comp}, \delta_h, \delta_g \rangle$ where δ_h and δ_g are closure derivations of h and g, respectively.

A *MinMax closure derivation* is a closure derivation relative to $\mathcal{P}_{\mathrm{MM}}$. *End of Definition 242.*

Definition 243 (Deterministic tie-breaking scheme)**.** For $d \in \mathbb{N}^{>0}$, a *(deterministic) tie-breaking selector of arity* d is a function

$$\rho_d : \mathcal{P}_+([1..d]) \to [1..d]$$

such that

$$\rho_d(J) \in J \qquad \forall J \in \mathcal{P}_+([1..d]).$$

A *(deterministic) tie-breaking scheme* is a family

$$\rho = \left(\rho_d\right)_{d \in \mathbb{N}^{>0}}$$

where ρ_d is a tie-breaking selector of arity d for every $d \in \mathbb{N}^{>0}$. We write

$$\rho[d] := \rho_d \qquad \forall d \in \mathbb{N}^{>0}.$$

End of Definition 243.

Definition 244 (Active variables of MinMax functions)**.** Let $f \in \mathcal{F}_{\mathrm{MM}}$ be a MinMax function,

$$f : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d_{\mathrm{out}}},$$

let δ be a MinMax closure derivation of f, and let ρ be a tie-breaking scheme.

Selected active-variable index map. We inductively define the *selected active-variable index map* of f relative to its closure derivation δ under the tie-breaking scheme ρ, denoted by

$$\sigma_\delta^\rho : \mathbb{R}^{d_{\mathrm{in}}} \to [1..d_{\mathrm{in}}]^{d_{\mathrm{out}}},$$

as follows:

- if $f = \pi_j^{d_{\mathrm{in}}}$ is the j-th coordinate projection,

$$\sigma_\delta^\rho(x) := j;$$

- if f is a Min or Max reduction, then

$$\sigma^{\rho}_{\delta}(x) := \rho_d\big(I_f(x)\big),$$

  where I_f is the active-variable index map of f as per Definition 225;

- if $f = (f_1, \ldots, f_k)$ and $\delta = \langle \mathsf{tupling}, \delta_1, \ldots, \delta_k \rangle$, then

$$\sigma^{\rho}_{\delta}(x) := \big(\sigma^{\rho}_{\delta_1}(x), \ldots, \sigma^{\rho}_{\delta_k}(x)\big);$$

- if $f = h \circ g$ and $\delta = \langle \mathsf{comp}, \delta_h, \delta_g \rangle$, then

$$\big(\sigma^{\rho}_{\delta}(x)\big)_i := \big(\sigma^{\rho}_{\delta_g}(x)\big)_{\phi(x;i)} \qquad \text{for} \qquad \phi(x;i) := \big(\sigma^{\rho}_{\delta_h}(g(x))\big)_i \qquad \forall i \in [1..d_{\text{out}}]\,.$$

We may write

$$\sigma^{\rho}_{\delta}(x;i) := \big(\sigma^{\rho}_{\delta}(x)\big)_i \qquad \forall i \in [1..d_{\text{out}}].$$

Active map. The *active map* of δ under the tie-breaking scheme ρ is the map $\mathrm{activeVariable}^{\rho}_{\delta}$ given by

$$\big(\mathrm{activeVariable}^{\rho}_{\delta}(x)\big)_i := (x)_j \qquad \text{for} \qquad j := \sigma^{\rho}_{\delta}(x;i) \qquad \forall i \in [1..d_{\text{out}}]\,.$$

We may write

$$\mathrm{activeVariable}^{\rho}_{\delta}(x;i) := \big(\mathrm{activeVariable}^{\rho}_{\delta}(x)\big)_i \qquad \forall i \in [1..d_{\text{out}}];$$

and then, when $d_{\text{out}} = 1$, we may also write

$$\mathrm{activeVariable}^{\rho}_{\delta}(x) := \mathrm{activeVariable}^{\rho}_{\delta}(x;1)\,.$$

When a deterministic tie-breaking scheme ρ and a MinMax derivation δ are clear from the context, we may omit them and hence simply write σ and $\mathrm{activeVariable}$ in place of σ^{ρ}_{δ} and $\mathrm{activeVariable}^{\rho}_{\delta}$, respectively. *End of Definition 244.*

Proposition 54 (Active-variable property of MinMax functions)**.** *Let $f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ be a MinMax function, let δ be a MinMax derivation of f, and let ρ be a deterministic tie-breaking scheme. Then*

$$(f(x))_\ell = \mathrm{activeVariable}^{\rho}_{\delta}(x;\ell) \qquad \forall x \in \mathbb{R}^{d_{\text{in}}},\ \forall \ell \in [1..d_{\text{out}}].$$

Proof. The claim follows by induction on the MinMax derivation δ.

For coordinate projections, it holds by definition. For Min and Max reductions, it follows from Definition 225. The tupling case follows coordinatewise.

Finally, suppose $f = h \circ g$, with corresponding derivations δ_h and δ_g. For every output coordinate $\ell \in [1..d_{\text{out}}]$, let

$$j := (\sigma^{\rho}_{\delta_h}(g(x)))_\ell.$$

By the induction hypothesis for h,

$$(f(x))_\ell = (h(g(x)))_\ell = (g(x))_j.$$

Applying the induction hypothesis for g to its j-th coordinate gives

$$(g(x))_j = \mathrm{activeVariable}^{\rho}_{\delta_g}(x;j).$$

By the definition of the active-variable function for composition,

$$\mathrm{activeVariable}^{\rho}_{\delta_g}(x;j) = \mathrm{activeVariable}^{\rho}_{\delta}(x;\ell),$$

which proves the claim. □

N.2 Piecewise Propagate-Reset Structure of MinMax Recurrences

N.2.1 Context and Structural Notions

Let us consider spaces $X \subseteq \mathbb{R}^n$, $U \subseteq \mathbb{R}^m$, and functions $R : U \to \mathbb{R}^{n\times n}$, $s : U \to \mathbb{R}^n$. Then, we introduce the corresponding MinMax dynamics function

$$f : X \times U \to \mathbb{R}^n, \qquad f(x,u) \;:=\; \big(R(u) \otimes x\big) \oplus s(u), \tag{156}$$

which can be written coordinate-wise as

$$\big(f(x,u)\big)_i \;=\; \left(\bigoplus_{j=1}^{n} \big(R(u)\big)_{i,j} \odot \big(x\big)_j\right) \oplus \big(s(u)\big)_i \qquad \forall i \in [1..n].$$

Definition 245 (State-affine dynamics function)**.** The considered dynamics function f is *state-affine* if there exist $A : U \to \mathbb{R}^{n\times n}$ and $b : U \to \mathbb{R}^n$ such that

$$f(x,u) = A(u) \cdot x + b(u) \qquad \forall x \in X,\ \forall u \in U.$$

Furthermore, it is *state-affine with input-independent linear coefficient* if A is constant; in this case, we call $\langle A, b\rangle$ a *linear-coefficient–offset-function pair for* f, or simply a *coefficient-offset pair for* f.

End of Definition 245.

Definition 246 (Propagate-reset state-affine map)**.** The dynamics function f is *propagate-reset* if it is state-affine with input-independent linear coefficient and additionally, for $\langle A, b\rangle$ its coefficient-offset pair, the following consditions hold jointly:

(1) A is deterministic row-substochastic;[18]

(2) the following implications hold:

$$(A)_{i,j} = 1 \quad\Longrightarrow\quad (b(u))_i = 0 \qquad \forall i,j \in [1..n],\ \forall u \in U.$$

End of Definition 246.

Remark 41. Every output coordinate produced by a propagate-reset map is either propagated from a previous state coordinate or set to the corresponding coordinate of its offset function. In particular, the two contributions are mutually exclusive: whenever $(A)_{i,:} \neq 0_n^\top$, we have $(b(u))_i = 0$.

End of Remark 41.

N.2.2 Source Variables and Active-Variable Regions

Definition 247 (Flat MinMax recurrence function)**.** The *flat MinMax recurrence function* is

$$\mathsf{mm} : \mathbb{R}^{2n+1} \to \mathbb{R}, \qquad \mathsf{mm}(z) \;:=\; (z)_1 \oplus \bigoplus_{j=1}^{n} \big((z)_{j+1} \odot (z)_{n+j+1}\big). \tag{157}$$

End of Definition 247.

Definition 248 (Source-variable label map)**.** For every $i \in [1..n]$, we define the *i-th source-variable label map*

$$\mathsf{sourceVariable}_i \;:\; [1..2n{+}1] \;\to\; \Big\{(s)_i\,,\,(R)_{i,1}\,,\ldots,\,(R)_{i,n}\,,\,(x)_1\,,\ldots,\,(x)_n\Big\}$$

by

$$\mathsf{sourceVariable}_i(k) := \begin{cases} (s)_i & \text{if } k = 1, \\ (R)_{i,j} & \text{if } k = 1{+}j \text{ and } j \in [1..n], \\ (x)_j & \text{if } k = 1{+}n{+}j \text{ and } j \in [1..n]. \end{cases}$$

End of Definition 248.

[18] The notion of *deterministic row-substochastic* matrix is introduced in Definition 21.

Definition 249 (MinMax source-variable maps)**.** For every $i \in [1..n]$, the *i-th source-variable map* is the function $\mathsf{sourceValue}_i : X \times U \to \mathbb{R}^{2n+1}$ given by

$$\mathsf{sourceValue}_i(x,u) := \Big(\big(s(u)\big)_i \quad \big(R(u)\big)_{i,1} \quad \cdots \quad \big(R(u)\big)_{i,n} \quad (x)_1 \quad \cdots \quad (x)_n \Big).$$

Then, the *source-variable maps* for the considered MinMax dynamics function f are $(\mathsf{sourceValue}_i)_{i=1}^n$. *End of Definition 249.*

Definition 250 (Active-variable pattern and partition of a MinMax recurrence)**.** Let mm be the flat MinMax recurrence function of Definition 247, let $(\mathsf{sourceValue}_i)_{i=1}^n$ be the source-variable maps of Definition 249, let δ be the MinMax closure derivation corresponding to the expression in (157), and let ρ be a tie-breaking scheme. Let us write $\sigma := \sigma_\delta^\rho$.

We define the *active-variable pattern map*

$$\alpha : X \times U \to [1..2n{+}1]^n, \qquad \alpha(x,u) := \Big(\sigma\big(\mathsf{sourceValue}_1(x,u)\big), \ldots, \sigma\big(\mathsf{sourceValue}_n(x,u)\big) \Big).$$

Then, writing

$$\hat{\alpha}(v) := \alpha^{-1}(V_v) \quad \text{for} \quad V_v := \{v\},$$

we define the corresponding *active-variable partition*

$$\mathcal{Z} := \Big\{ \hat{\alpha}(v) \;\Big|\; v \in [1..2n{+}1]^n,\ \hat{\alpha}(v) \neq \emptyset \Big\}.$$

We call every $Z \in \mathcal{Z}$ an *active-variable region* of f induced by δ and ρ. *End of Definition 250.*

Fact 52 (Finiteness of the active-variable partition)**.** *The set $\mathcal{Z}$ of Definition 250 is a finite partition of $X \times U$. In particular, its cardinality is bounded as*

$$\big|\mathcal{Z}\big| \leq (2n{+}1)^n.$$

End of Fact 52.

Proof. The active-variable pattern map

$$\alpha : X \times U \to [1..2n{+}1]^n$$

has finite codomain of cardinality $(2n{+}1)^n$. Its nonempty fibres are pairwise disjoint and cover $X \times U$. By definition, these nonempty fibres are precisely the elements of $\mathcal{Z}$. □

Definition 251 (Regional active source variable)**.** Let α and $\mathcal{Z}$ be the active-variable pattern map and active-variable partition of Definition 250, respectively.

Let $Z \in \mathcal{Z}$.

Since Z is an active-variable region, there exists a unique

$$a^Z \in [1..2n{+}1]^n$$

such that

$$\alpha(x,u) = a^Z \qquad \forall (x,u) \in Z.$$

For every $i \in [1..n]$, we define

$$a_i^Z := (a^Z)_i$$

and the *active source variable of f for coordinate i on Z*

$$\mathsf{activeSourceVariable}(f; Z, i) := \mathsf{sourceVariable}_i(a_i^Z).$$

End of Definition 251.

Fact 53 (Coordinate representation of MinMax recurrences)**.** *Let* mm *be the flat MinMax recurrence function of Definition 247, and let* $(\mathsf{sourceValue}_i)_{i=1}^n$ *be the source-variable maps of Definition 249. Then,*

$$\big(f(x,u)\big)_i \;=\; \mathsf{mm}\big(\mathsf{sourceValue}_i(x,u)\big) \qquad \forall i \in [1..n],\ \forall (x,u) \in X \times U.$$

End of Fact 53.

Proof. By Definitions 247 and 249, for every $i \in [1..n]$,

$$\mathsf{mm}\big(\mathsf{sourceValue}_i(x,u)\big) \;=\; (s(u))_i \oplus \bigoplus_{j=1}^{n} \big((R(u))_{i,j} \odot (x)_j\big) \;=\; (f(x,u))_i,$$

where the last equality follows from Equation (156). □

N.2.3 Piecewise Propagate-Reset Representation

Proposition 55 (Piecewise propagate-reset representation of MinMax recurrences)**.** *Let $\mathcal{Z}$ be the active-variable partition of Definition 250.*

There exists a piecewise representation $\gamma = \{\langle Z, f_Z\rangle \mid Z \in \mathcal{Z}\}$ on $(X \times U, \mathbb{R}^n)$ such that:

(1) γ has input partition $\mathcal{Z}$;

(2) γ represents f;

(3) γ is affine in the first argument;

(4) every piece f_Z is a propagate-reset map.

Equivalently, there exists a coefficient-based piecewise representation $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ on $(X, U, \mathbb{R}^n)$ that is affine in the first argument, represents f, and satisfies the following property: for every $Z \in \mathcal{Z}$, the coefficient-offset pair $\langle A_Z, b_Z\rangle$ defines a propagate-reset map

$$f_Z(x,u) := A_Z \cdot x + b_Z(u). \tag{158}$$

The representation γ can be chosen as the canonical piecewise representation induced by $\hat{\gamma}$; in particular,

$$\check{\gamma} = \check{\hat{\gamma}} = f.$$

Proof. By Fact 52, $\mathcal{Z}$ is a finite partition of $X \times U$.

Fix $Z \in \mathcal{Z}$. For every $i \in [1..n]$, let a_i^Z be the regional active-variable index of Definition 251.

By Definitions 250 and 251,

$$\sigma_\delta^\rho\big(\mathsf{sourceValue}_i(x,u)\big) = a_i^Z \qquad \forall (x,u) \in Z.$$

By Fact 53 and Proposition 54, we have

$$\begin{aligned}(f(x,u))_i &= \mathsf{mm}\big(\mathsf{sourceValue}_i(x,u)\big)\\ &= \mathrm{activeVariable}_\delta^\rho\big(\mathsf{sourceValue}_i(x,u); 1\big)\\ &= (\mathsf{sourceValue}_i(x,u))_{a_i^Z} \qquad \forall (x,u) \in Z.\end{aligned} \tag{159}$$

We now define

$$A_Z \in \mathbb{R}^{n\times n}$$

and

$$b_Z : U \to \mathbb{R}^n$$

row by row. For every $i \in [1..n]$, set

$$((A_Z)_{i,:}, (b_Z(u))_i) := \begin{cases} (0_n^\top, (s(u))_i), & a_i^Z = 1,\\ (0_n^\top, (R(u))_{i,j}), & a_i^Z = 1{+}j, \quad j \in [1..n],\\ (e_j^\top, 0), & a_i^Z = 1{+}n{+}j, \quad j \in [1..n]. \end{cases} \tag{160}$$

Since

$$a_i^Z \in [1..2n{+}1],$$

exactly one of these cases applies.

Define

$$f_Z : X \times U \to \mathbb{R}^n, \qquad f_Z(x,u) := A_Z \cdot x + b_Z(u).$$

By Equation (159) and Definition 249, the i-th coordinate of $f_Z(x,u)$ is precisely

$$(\mathsf{sourceValue}_i(x,u))_{a_i^Z}$$

for every $(x,u) \in Z$. Hence, coordinatewise,

$$f(x,u) = f_Z(x,u) \qquad \forall (x,u) \in Z.$$

Therefore, $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ is a coefficient-based piecewise representation on $(X, U, \mathbb{R}^n)$ that is affine in the first argument. By its semantics,

$$\check{\hat{\gamma}} = f.$$

By Fact 9.2, $\hat{\gamma}$ induces the canonical piecewise representation $\gamma = \{\langle Z, f_Z\rangle \mid Z \in \mathcal{Z}\}$ which is affine in the first argument and satisfies

$$\check{\gamma} = \check{\hat{\gamma}} = f.$$

It remains to show that every f_Z is propagate-reset. By Equation (160), every row of A_Z is either $0_n^\top$ or a standard basis row $e_j^\top$. Hence A_Z is deterministic row-substochastic.

Furthermore, if

$$(A_Z)_{i,j} = 1,$$

then the third case of Equation (160) applies to row i, and therefore

$$(b_Z(u))_i = 0 \qquad \forall u \in U.$$

Thus $\langle A_Z, b_Z\rangle$ satisfies the propagate-reset conditions of Definition 246.

Therefore every piece f_Z is a propagate-reset map, completing the proof. □

N.2.4 State-Coordinate Behaviours

Definition 252 (State-propagation behaviour)**.** Let $Z \in \mathcal{Z}$ be an active-variable region, and let $i \in [1..n]$. We say that the i-th state coordinate exhibits *state-propagation behaviour through f on Z* if there exists $j \in [1..n]$ such that

$$\mathsf{activeSourceVariable}(f; Z, i) = (x)_j.$$

In this case, the j-th previous state coordinate is the *propagated state coordinate*.

End of Definition 252.

Fact 54 (Characterisation of state-propagation behaviour)**.** *Consider the coefficient-based piecewise representation $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ of Proposition 55. Suppose that the i-th state coordinate exhibits state-propagation behaviour through f on Z with propagated state coordinate $j \in [1..n]$.*

- *First, we have*

$$(f(x,u))_i = (x)_j \qquad \forall (x,u) \in Z.$$

- *Second, we have*

$$(A_Z)_{i,:} = e_j^\top \qquad \textit{and} \qquad (b_Z(u))_i = 0 \qquad \forall u \in U.$$

- *Third, the propagated state value satisfies*

$$(s(u))_i \le (x)_j \le (R(u))_{i,j} \qquad \forall (x,u) \in Z. \tag{161}$$

End of Fact 54.

Proof. By Definition 251,

$$\mathsf{activeSourceVariable}(f; Z, i) = \mathsf{sourceVariable}_i(a_i^Z).$$

Hence, by Definition 248,

$$\mathsf{activeSourceVariable}(f; Z, i) = (x)_j \qquad \Longrightarrow \qquad a_i^Z = 1{+}n{+}j.$$

Therefore, Equation (159) gives

$$(f(x,u))_i = (x)_j \qquad \forall (x,u) \in Z.$$

Furthermore, Equation (160) gives

$$(A_Z)_{i,:} = e_j^\top ,$$
$$(b_Z(u))_i = 0 \qquad \forall u \in U.$$

It remains to prove Equation (161). Since $(x)_j$ is the active source variable, it is selected by the inner Min operation

$$(R(u))_{i,j} \odot (x)_j.$$

Hence

$$(x)_j \leq (R(u))_{i,j}.$$

The resulting value is subsequently selected by the outer Max operation, whose arguments include $(s(u))_i$. Hence

$$(s(u))_i \leq (x)_j.$$

Combining the two inequalities proves Equation (161). □

Definition 253 (State-reset behaviour)**.** Let $Z \in \mathcal{Z}$ be an active-variable region, and let $i \in [1..n]$.

We say that the i-th state coordinate exhibits *state-reset behaviour through f on Z* if

$$\mathsf{activeSourceVariable}(f; Z, i) \in \Big\{(s)_i, (R)_{i,1}, \ldots, (R)_{i,n}\Big\}.$$

End of Definition 253.

Fact 55 (Characterisation of state-reset behaviour)**.** *Consider the coefficient-based piecewise representation $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ of Proposition 55.*

Suppose that the i-th state coordinate exhibits state-reset behaviour through f on Z. Then

$$(A_Z)_{i,:} = 0_n^\top,$$

and exactly one of the following two cases holds:

- *we have*

  $$\mathsf{activeSourceVariable}(f; Z, i) = (s)_i,$$

  in which case

  $$(f(x,u))_i = (s(u))_i \qquad \forall (x,u) \in Z,$$
  $$(b_Z(u))_i = (s(u))_i \qquad \forall u \in U;$$

- *there exists $j \in [1..n]$ such that*

  $$\mathsf{activeSourceVariable}(f; Z, i) = (R)_{i,j},$$

  in which case

  $$(f(x,u))_i = (R(u))_{i,j} \qquad \forall (x,u) \in Z,$$
  $$(b_Z(u))_i = (R(u))_{i,j} \qquad \forall u \in U.$$

End of Fact 55.

Remark 42. Thus, under state-reset behaviour, the updated state coordinate is independent of the previous state and is set directly to an input-dependent source value. *End of Remark 42.*

Proof. By Definition 251,

$$\mathsf{activeSourceVariable}(f; Z, i) = \mathsf{sourceVariable}_i(a_i^Z).$$

Suppose first that

$$\mathsf{activeSourceVariable}(f; Z, i) = (s)_i.$$

By Definition 248, we have

$$a_i^Z = 1.$$

Therefore, Equations (159) and (160) yield

$$(f(x,u))_i = (s(u))_i \qquad \forall (x,u) \in Z,$$
$$(A_Z)_{i,:} = 0_n^\top, \qquad (b_Z(u))_i = (s(u))_i \qquad \forall u \in U.$$

Otherwise, state-reset behaviour implies that there exists $j \in [1..n]$ such that

$$\mathsf{activeSourceVariable}(f; Z, i) = (R)_{i,j}.$$

By Definition 248,

$$a_i^Z \; = \; 1{+}j.$$

Therefore, Equations (159) and (160) yield

$$(f(x,u))_i = (R(u))_{i,j} \qquad \forall (x,u) \in Z,$$
$$(A_Z)_{i,:} = 0_n^\top,$$
$$(b_Z(u))_i = (R(u))_{i,j} \qquad \forall u \in U.$$

□

Proposition 56 (State propagation–reset dichotomy)**.** *For every $Z \in \mathcal{Z}$ and every $i \in [1..n]$, exactly one of the following two behaviours occurs:*

(1) the i-th state coordinate exhibits state-propagation behaviour through f on Z;

(2) the i-th state coordinate exhibits state-reset behaviour through f on Z.

End of Proposition 56.

Proof. Fix $Z \in \mathcal{Z}$ and $i \in [1..n]$.

By Definition 251,

$$\mathsf{activeSourceVariable}(f; Z, i) = \mathsf{sourceVariable}_i(a_i^Z).$$

By Definition 248, exactly one of the following holds:

$$\mathsf{activeSourceVariable}(f; Z, i) \in \Big\{(s)_i, (R)_{i,1}, \ldots, (R)_{i,n}\Big\},$$

or there exists $j \in [1..n]$ such that

$$\mathsf{activeSourceVariable}(f; Z, i) = (x)_j.$$

By Definitions 253 and 252, these alternatives are precisely state-reset behaviour and state-propagation behaviour, respectively. □

N.2.5 Regularity Consequences

Fact 56 (Piecewise-affinity of MinMax state transformations)**.** *For every $u \in U$, the state transformation $f_u(x) := f(x,u)$ is piecewise affine.* End of Fact 56.

Proof. By Proposition 55, the dynamics function f admits a piecewise representation that is affine in its first argument. The claim therefore follows directly from Fact 10. □

Proposition 57 (Piecewise-affine regularity of MinMax recurrences)**.** *Assume that the reset R and set s functions are piecewise affine.*

Then, the MinMax dynamics function f is piecewise affine; namely, for $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ the coefficient-based piecewise propagate-reset representation of Proposition 55, there exists a piecewise-affine coefficient-based representation

$$\hat{\gamma}_f := \big\{\langle Q, C_Q, c_Q\rangle\big\}_{Q \in \mathcal{Q}}$$

representing f such that $\mathcal{Q} \preceq \mathcal{Z}$.

For every $Q \in \mathcal{Q}$, let $Z_Q \in \mathcal{Z}$ be the unique active-variable region satisfying $Q \subseteq Z_Q$. Then there exists $B_Q \in \mathbb{R}^{n\times m}$ such that

$$C_Q = \begin{pmatrix} A_{Z_Q} & B_Q \end{pmatrix},$$

and

$$f(x,u) \;=\; A_{Z_Q} \cdot x + B_Q \cdot u + c_Q \qquad \forall (x,u) \in Q.$$

Remark 43. Thus, all regions Q refining the same active-variable region Z share the same state-linear coefficient A_Z. *End of Remark 43.*

Proof. Since R and s are piecewise affine, each of their finitely many scalar coordinate functions

$$u \mapsto (s(u))_i, \qquad u \mapsto (R(u))_{i,j}$$

is piecewise affine. Taking a common refinement of the corresponding finite input partitions yields a finite partition $\mathcal{P}$ of U such that all these scalar functions are affine on every $P \in \mathcal{P}$.

Define

$$\mathcal{Q} := \{ Z \cap (X \times P) \mid Z \in \mathcal{Z},\ P \in \mathcal{P},\ Z \cap (X \times P) \neq \emptyset \}.$$

This is a finite partition of $X \times U$ satisfying $\mathcal{Q} \preceq \mathcal{Z}$.

Fix

$$Q = Z \cap (X \times P) \in \mathcal{Q}.$$

By Proposition 55,

$$f(x,u) = A_Z \cdot x + b_Z(u) \qquad \forall (x,u) \in Z.$$

By Equation (160), every coordinate of b_Z is either identically zero or equal to one of

$$u \mapsto (s(u))_i, \qquad u \mapsto (R(u))_{i,j}.$$

Hence b_Z is affine on P. Therefore there exist

$$B_Q \in \mathbb{R}^{n\times m}, \qquad c_Q \in \mathbb{R}^n$$

such that

$$b_Z(u) = B_Q \cdot u + c_Q \qquad \forall u \in P.$$

Consequently,

$$f(x,u) = A_Z \cdot x + B_Q \cdot u + c_Q \qquad \forall (x,u) \in Q.$$

Define

$$C_Q := \begin{bmatrix} A_Z & B_Q \end{bmatrix}.$$

Then

$$\hat{\gamma}_f := \left\{ \langle Q, C_Q, c_Q \rangle \right\}_{Q \in \mathcal{Q}}$$

is a piecewise-affine coefficient-based representation of f. By Fact 7.2, it induces a canonical piecewise-affine representation of f. Hence f is piecewise affine.

Finally,

$$C_Q = \begin{bmatrix} A_Z & B_Q \end{bmatrix}$$

shows that all refined regions contained in the same Z share the same state-linear coefficient A_Z. □

Proposition 58 (PAP regularity of MinMax recurrences)**.** *Suppose that the reset R and set s functions are PAP. Then the dynamics function f is PAP. In particular, if R and s are analytic, then f is PAP.*

Proof. Let mm be the flat MinMax recurrence function of Definition 247, and let $(\mathsf{sourceValue}_i)_{i=1}^n$ be the source-variable maps of Definition 249.

Coordinate projections are analytic and hence PAP. Since PAP functions are closed under pointwise minimum and maximum, Equation (157) implies that mm is PAP.

Fix $i \in [1..n]$. By Definition 249,

$$\mathsf{sourceValue}_i(x,u) = \Big(\big(s(u)\big)_i, \big(R(u)\big)_{i,1}, \ldots, \big(R(u)\big)_{i,n}, (x)_1, \ldots, (x)_n\Big).$$

Its state-coordinate components are analytic and hence PAP, while its remaining components are PAP by hypothesis. Closure under finite tupling therefore implies that $\mathsf{sourceValue}_i$ is PAP.

Since PAP maps are closed under composition, $\mathsf{mm} \circ \mathsf{sourceValue}_i$ is PAP. By Fact 53,

$$(f(x,u))_i = \mathsf{mm}\big(\mathsf{sourceValue}_i(x,u)\big),$$

so every coordinate of f is PAP. Closure under finite tupling then implies that f is PAP.

Finally, every analytic map is PAP, which proves the last claim. □

N.2.6 Stability Consequences

Theorem 52 (State non-expansiveness)**.** *Let $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ be the coefficient-based piecewise propagate-reset representation of Proposition 55. Then,*

$$\|A_Z\|_\infty \le 1 \qquad \forall Z \in \mathcal{Z}.$$

Furthermore,

$$\|f(x,u) - f(x',u)\|_\infty \le \|x - x'\|_\infty \qquad \forall u \in U,\ \forall x \in X,\ \forall x' \in X. \tag{162}$$

Proof. By Proposition 55, every A_Z is deterministic row-substochastic. Hence

$$\|A_Z\|_\infty \le 1.$$

For the global bound, fix $u \in U$. For every $i, j \in [1..n]$, the scalar map

$$z \mapsto \min\{(R(u))_{i,j}, z\}$$

is 1-Lipschitz. Furthermore, the maximum of finitely many scalar arguments is 1-Lipschitz with respect to the ℓ_∞ norm of those arguments. Applying these facts coordinatewise to Equation (156) gives Equation (162). □

Remark 44. Proposition 56 and Equation (161) explain the strong one-step bound used in the uBIBS analysis—see §L.1:

- under *state-reset behaviour*, the updated coordinate is set directly to a current input-dependent source value;
- under *state-propagation behaviour*, the propagated state value is bounded above and below by current input-dependent source values.

Thus, in either behaviour, the updated state can be controlled in terms of current input-dependent quantities rather than through additive accumulation of the previous state. *End of Remark 44.*

N.3 Composition and Relation to Linear Recurrences

The propagate-reset coefficient structure is preserved under composition.

Proposition 59 (Closure of propagate-reset structure under composition)**.** *For $k \in \{1, 2\}$, let*

$$F_k(x) \;=\; A_k \cdot x + b_k$$

be an affine map such that A_k is deterministic row-substochastic and

$$(A_k)_{i,j} = 1 \quad \Longrightarrow \quad (b_k)_i = 0 \qquad \forall i, j \in [1..n].$$

Then,

$$F_2 \circ F_1(x) \;=\; A \cdot x + b \qquad \textit{for} \qquad A \;:=\; A_2 \cdot A_1, \quad b \;:=\; A_2 \cdot b_1 + b_2,$$

satisfies the same propagate-reset coefficient conditions.

Proof. Fix row i. If $(A_2)_{i,\cdot} = 0_n^\top$, then $A_{i,\cdot} = 0_n^\top$ and $b_i = (b_2)_i$. Otherwise $(A_2)_{i,\cdot} = e_k^\top$, and hence $(b_2)_i = 0$. If $(A_1)_{k,\cdot} = 0_n^\top$, then $A_{i,\cdot} = 0_n^\top$ and $b_i = (b_1)_k$; if $(A_1)_{k,\cdot} = e_j^\top$, then $(b_1)_k = 0$ and $A_{i,\cdot} = e_j^\top$, $b_i = 0$. □

Definition 254 (Active-variable regions and coefficients along a MinMax trajectory)**.** Let $\mathcal{Z}$ be the partition of Definition 250, and let $\hat{\gamma} = \{\langle Z, A_Z, b_Z\rangle \mid Z \in \mathcal{Z}\}$ be the coefficient-based piecewise propagate-reset representation of f given by Proposition 55.

Assume

$$f(x,u) \in X \qquad \forall (x,u) \in X \times U.$$

Let $u = (u_t)_{t\in\mathbb{T}} \in U^\omega$, let $x_0 \in X$, and let $(x_t)_{t\in\mathbb{T}}$ be the corresponding MinMax trajectory,

$$x_t \;:=\; f(x_{t-1}, u_t) \qquad \forall t \in \mathbb{T}.$$

For every $t \in \mathbb{T}$, let $Z_t \in \mathcal{Z}$ be the unique active-variable region such that $(x_{t-1}, u_t) \in Z_t$. We call $(Z_t)_{t\in\mathbb{T}}$ the *active-region sequence* along the trajectory $(x_t)_{t\in\mathbb{T}}$ induced by u. For every $t \in \mathbb{T}$, we define

$$A_t := A_{Z_t}, \qquad b_t := b_{Z_t}(u_t).$$

We call $(A_t)_{t\in\mathbb{T}}$ and $(b_t)_{t\in\mathbb{T}}$ the *active coefficient sequences* along the trajectory $(x_t)_{t\in\mathbb{T}}$ induced by u. *End of Definition 254.*

Fact 57. *In the context of Definition 254, we have*

$$x_t \;=\; A_t \cdot x_{t-1} + b_t \qquad \forall t \in \mathbb{T}.$$

End of Fact 57.

Proposition 60 (Single-source propagation and no additive accumulation)**.** *Let $u = (u_t)_{t\in\mathbb{T}} \in U^\omega$, let $x_0 \in X$, and let $(x_t)_{t\in\mathbb{T}}$, $(A_t)_{t\in\mathbb{T}}$, and $(b_t)_{t\in\mathbb{T}}$ be the corresponding state and active coefficient sequences of Definition 254. Let $s, t \in \mathbb{N}^{>0}$ with $s < t$. Then, the following hold.*

- *There exists $\langle A_{t:s}, b_{t:s}\rangle$ such that*

  $$x_t \;=\; A_{t:s} \cdot x_s + b_{t:s},$$

  and $\langle A_{t:s}, b_{t:s}\rangle$ satisfies the propagate-reset coefficient conditions

  $$\begin{aligned} &A_{t:s} \in \{0,1\}^{n\times n},\\ &A_{t:s} \cdot 1_n \;\le\; 1_n,\\ &(A_{t:s})_{i,j} = 1 \implies (b_{t:s})_i = 0 \qquad \forall i,j \in [1..n]. \end{aligned}$$

- *Every coordinate of $b_{t:s}$ is either zero or a single coordinate of an active offset introduced at some time point $\tau \in [s{+}1..t]$. Namely, for every $i \in [1..n]$, at least one of Equations (163) and (164) holds:*

  $$(b_{t:s})_i = 0, \tag{163}$$

  $$(b_{t:s})_i \in \mathrm{entries}(b_\tau) \qquad \text{for some } \tau \in [s{+}1..t]. \tag{164}$$

- *Every coordinate of x_t comes from*

  (i) a single coordinate of x_s, or
  (ii) a single input-dependent source value introduced at some time point $\tau \in [s{+}1..t]$;

  namely, for every $i \in [1..n]$, at least one of the conditions given by Equations (165)–(167) holds:

  $$(x_t)_i \in \mathrm{entries}(x_s)\,, \tag{165}$$

  $$(x_t)_i \in \mathrm{entries}(R(u_\tau)) \qquad \text{for some } \tau \in [s{+}1..t], \tag{166}$$

  $$(x_t)_i \in \mathrm{entries}(s(u_\tau)) \qquad \text{for some } \tau \in [s{+}1..t]. \tag{167}$$

- *State values admit the bound*

  $$\|x_t\|_\infty \;\le\; \max\big\{\, \|x_s\|_\infty,\, \|b_{s+1}\|_\infty,\, \|b_{s+2}\|_\infty,\, \ldots,\, \|b_t\|_\infty \,\big\}.$$

Proof. By Definition 254, for every $\tau \in \mathbb{T}$,

$$A_\tau \;=\; A_{Z_\tau}, \qquad b_\tau \;=\; b_{Z_\tau}(u_\tau),$$

where $\langle A_{Z_\tau}, b_{Z_\tau} \rangle$ are the coefficients of the piecewise propagate-reset representation fixed in that definition. Hence, $\langle A_\tau, b_\tau \rangle$ satisfies the corresponding propagate-reset coefficient conditions.

Repeated application of Proposition 59 to the coefficient pairs

$$\langle A_{s+1}, b_{s+1} \rangle, \ldots, \langle A_t, b_t \rangle$$

yields a pair

$$\langle A_{t:s}, b_{t:s} \rangle \in \mathbb{R}^{n \times n} \times \mathbb{R}^n$$

satisfying the propagate-reset coefficient conditions and such that

$$x_t \;=\; A_{t:s} \cdot x_s + b_{t:s}.$$

This proves the first claim.

Furthermore, Proposition 59 implies that, under each composition, every coordinate of the accumulated offset either remains zero, propagates a single previously introduced offset coordinate, or is replaced by a single coordinate of the newly composed offset. Consequently, no two offset coordinates introduced at distinct time points are ever added. It follows inductively that, for every $i \in [1..n]$,

$$(b_{t:s})_i = 0 \qquad \text{or} \qquad (b_{t:s})_i \in \mathrm{entries}(b_\tau)$$

for some $\tau \in [s{+}1..t]$. This proves the second claim.

By the form of the active regional affine representation, for every $\tau \in [s{+}1..t]$ and every $i \in [1..n]$, either the i-th row of A_τ contains exactly one entry equal to one and

$$(b_\tau)_i \;=\; 0,$$

or the i-th row of A_τ is zero and

$$(b_\tau)_i \in \mathrm{entries}(R(u_\tau)) \,\cup\, \mathrm{entries}(s(u_\tau))\,.$$

Hence, at each time step and in each coordinate, the update either propagates a single state coordinate or replaces it by a single input-dependent source value.

Since this propagate-reset structure is preserved under composition, for every $i \in [1..n]$, the coordinate $(x_t)_i$ is either a single coordinate of x_s, or a single input-dependent source value introduced at some time $\tau \in [s{+}1..t]$. Therefore,

$$(x_t)_i \in \mathrm{entries}(x_s) \,\cup\, \bigcup_{\tau=s+1}^{t} \left(\mathrm{entries}(R(u_\tau)) \,\cup\, \mathrm{entries}(s(u_\tau))\right).$$

Combining this observation with the preceding characterization gives Equations (165)–(167).

Finally, every coordinate of x_t is therefore equal to either a coordinate of x_s or a coordinate of some b_τ with $\tau \in [s{+}1..t]$. Hence

$$\|x_t\|_\infty \;\le\; \max\left\{\, \|x_s\|_\infty,\, \|b_{s+1}\|_\infty,\, \ldots,\, \|b_t\|_\infty \,\right\},$$

which proves the last claim. □

Remark 45 (Comparison with linear recurrences). Proposition 60 sharply contrasts MinMax with a generic linear recurrence

$$x_t \;=\; A_t \cdot x_{t-1} + b_t.$$

Indeed, unrolling a generic linear recurrence gives

$$x_t \;=\; A_t \cdots A_1 \cdot x_0 \;+\; \sum_{\tau=1}^{t} A_t \cdots A_{\tau+1} \cdot b_\tau.$$

Hence, bounded additive contributions may accumulate along non-contracting directions. For example, the recurrence

$$x_t = x_{t-1} + 1$$

is non-expansive in the state variable but is not BIBS stable. In contrast, Proposition 60 shows that, for MinMax, the corresponding additive expansion collapses coordinatewise to a single surviving source value: contributions introduced at different time points do not add coordinatewise.

The comparison with contractive linear recurrences is also instructive. A condition such as

$$\|A_t\| \leq \alpha < 1$$

ensures stability through geometric attenuation of past contributions. MinMax instead only guarantees

$$\|A_t\|_\infty \leq 1$$

and may therefore preserve a state value with unit gain indefinitely. Its stability mechanism is not geometric attenuation, but single-source propagation and replacement without additive accumulation, together with the bracketing property in (161).

The analogy with linear recurrences is nevertheless only local. The active coefficient matrix A_t depends on the active-variable region containing (x_{t-1}, u_t) and therefore generally depends on the current state. Thus, MinMax is a switched piecewise state-affine system rather than a globally linear recurrent system. *End of Remark 45.*

Part IV

Implementation and Experiments

O Implementation of MinMax RNCs

Our implementation of MinMax RNCs of state-degree one is released as a Python package [63] with source code available at [44], and for MinMax RNCs of arbitrary-degree we use a more prototypical implementation that follows the same principles—details in the extended version [64].

We next describe our implementation of MinMax RNCs of state-degree one, version `v0.3.0`, used in the experiments. The overall architecture is below: `MinMaxNeuron` corresponds to a layer in the formal description; and `MinMaxLayer` is such neuron equipped with convolution, feedforward network, normalisation, and residual connections.

```
MinMaxRNC_LM
├─ token_emb :  Embedding(vocab_size, d_model)
├─ MinMaxRNC : backbone
│  ├─ MinMaxLayer × n_layers
│  │  ├─ norm_conv → BasicConv / GatedConv
│  │  ├─ norm_ffn → FeedForward / GatedFeedForward
│  │  └─ norm_neuron → MinMaxNeuron
│  ├─ postlayers_ffn (optional)
│  └─ postlayers_norm
├─ head_drop :  Dropout
└─ lm_head :  Linear(d_model, vocab_size)
```

O.1 MinMax Neuron

A `MinMaxNeuron` is the recurrent cell. It projects the input to a vector $\mathbf{r}_t$ and a vector $\mathbf{s}_t$, runs the parallel MinMax scan element-wise, and projects the resulting hidden states back to the residual-stream dimension.

The state $\mathbf{x}_t \in \mathbb{R}^D$ is updated elementwise as:

$$\mathbf{x}_{t+1} \;=\; (\mathbf{r} \odot \mathbf{x}_t) \oplus \mathbf{s}_t,$$

where $\mathbf{r}$ and $\mathbf{s}$ are linear projections of the input $\mathbf{u} \in \mathbb{R}^I$:

$$\begin{aligned} \mathbf{r}_t &= W_{\mathrm{r}} \cdot \mathbf{u}_t \\ \mathbf{s}_t &= W_{\mathrm{s}} \cdot \mathbf{u}_t \end{aligned}$$

The output is projected back to the residual-stream dimension:

$$\mathbf{y}_t = \begin{cases} W_{\mathrm{o}} \cdot \mathbf{x}_t & \text{if } \texttt{output_gate=False}, \\ W_{\mathrm{o}} \cdot (\mathbf{x}_t * \mathrm{sigmoid}(W_{\mathrm{g}} \cdot \mathbf{u}_t)) & \text{if } \texttt{output_gate=True}, \end{cases}$$

where $*$ is element-wise product, and the learnable parameters are W_{r}, W_{s}, W_{o}, W_{g} (and optionally $\mathbf{x}_0$).

All states for a sequence of length T are computed simultaneously via a parallel prefix scan in $O(\log T)$ depth. At inference time the recurrence runs sequentially with $O(1)$ compute and $O(D)$ memory per token.

O.2 MinMax Layer

A `MinMaxLayer` is the residual block. The data flow uses pre-norm; only the neuron output is added back to the residual stream:

$$\begin{aligned} \textit{conv} &= \mathrm{Conv}\big(\mathrm{norm}(\mathbf{u})\big), \\ \textit{ffn} &= \mathrm{FFN}\big(\mathrm{norm}(\mathbf{u} + \mathrm{conv})\big), \\ \textit{neuron} &= \mathrm{Neuron}\big(\mathrm{norm}(\mathbf{u} + \textit{conv} + \textit{ffn})\big), \\ \textit{output} &= \mathbf{u} + \textit{neuron}. \end{aligned}$$

The Conv and FFN outputs feed the neuron's input but are not added to the residual stream independently. The convolution provides short-range context (one previous token). The FFN mixes features. The neuron integrates information over arbitrarily long ranges via the recurrence.

O.2.1 Convolution Variants

The implementation provides two convolution variants:

- `BasicConv` (default): a learned linear projection that concatenates $[\mathbf{u}_{t-1},\ \mathbf{u}_t]$ and maps back to $\mathbb{R}^D$, giving more expressive local mixing at the cost of $2\times$ the parameters.
- `GatedConv`: a learned per-feature scalar gate $\mathbf{g} \in \mathbb{R}^D$ interpolates between the previous and current token:

$$\mathrm{out}_t \ = \ \mathrm{sigmoid}(\mathbf{g}) * \mathbf{u}_{t-1} \ + \ \big(1 - \mathrm{sigmoid}(\mathbf{g})\big) * \mathbf{u}_t.$$

O.3 MinMax RNC

The overall architecture `MinMaxRNC` is obtained by stacking `n_layers` MinMaxLayers. An optional post-layers FFN and normalisation are applied after the final layer.

All architecture hyperparameters are specified through a single flat `MinMaxRNCConfig` dataclass described in Table 2.

O.4 Language Model Wrapper

`MinMaxRNC_LM` wraps `MinMaxRNC` adding:

- a token embedding (vocab_size $\times$ d_{model}) initialised with `small_init_` (see Section O.5 below).
- a dropout before the LM head.
- a linear LM head ($d_{\text{model}} \times$ vocab_size), optionally tied to the embedding weights.

All architecture hyperparameters are specified through a single flat `MinMaxRNCLMConfig` dataclass described in Table 3.

O.5 Initialisation

The following initialisation scheme is employed:

- `small_init_` [65] is applied to all *input-side* projections: token embedding, LM head (when untied), BasicConv, FFN up-projection, and the neuron W_s, W_r, and output-gate W_g projections. It keeps activations small at initialisation in deep networks.

$$\texttt{small_init_}(z, \mathrm{dim}): \qquad z \sim \mathcal{N}(0, \sigma^2),\ \sigma = \sqrt{2/(5 \cdot \mathrm{dim})}$$

- `wang_init_` [6, 66] is applied to the two *residual output* projections: the neuron output W_{o} and the FFN down-projection. It additionally divides by n_{layers} (the number of residual blocks) so the total variance contributed by all residual branches to the stream stays $O(1/n_{\text{layers}})$ regardless of depth, following the GPT-2 scaled-init strategy.

$$\texttt{wang_init_}(z, \mathrm{dim}, n_{\text{layers}}): \qquad z \sim \mathcal{N}(0, \sigma^2),\ \sigma = 2/(n_{\text{layers}} \cdot \sqrt{\mathrm{dim}})$$

- for the s and r projections, three initialisation types are possible, as described in the following table.

Initialisation	W_s **weights**	W_r **weights**	b_s	b_r
`'small_init'`	`small_init_`	`small_init_`	0	0
`'kaiming'`	Kaiming uniform ($a = \sqrt{5}$)	Kaiming uniform ($a = \sqrt{5}$)	0	0
`'asymmetric'`	Kaiming uniform ($a = \sqrt{5}$)	`small_init_`	$+1$	0

The `asymmetric` scheme addresses an initialisation dead zone: with $\mathbf{x}_0 = \mathbf{0}$ and both projections near zero, the recurrence $(\mathbf{r}_t \odot \mathbf{x}_{t-1}) \oplus \mathbf{s}_t$ keeps every hidden unit at 0 whenever $s < 0 < r$, preventing state writes early in training. Setting $b_s = +1$ ensures $s > 0$ at initialisation so writes happen from the first token; keeping W_r small prevents r from immediately over-writing the written values.

O.6 Feedforward Networks

The implementation provides two variants of feedforward networks:

- **Gated variant** (default): the gated variant is the default (`ffn_type='gated'`). With activation ϕ =ReLU it is ReGLU (default); with ϕ =Swish it is SwiGLU.
- **Basic variant**: one-hidden layer MLP (`ffn_type='basic'`).

Class	**Formula**	**Reference**
`FeedForward`	$W_2 \cdot \phi(W_1 \cdot \mathbf{x})$	Cf. [1]
`GatedFeedForward`	$W_2 \cdot (\phi(\text{gate}) * \text{value})\,,\ [\text{gate} \mid \text{value}] = W_1\mathbf{x}$	Cf. [67, 68]

Table 2: Description of the `MinMaxRNCConfig` dataclass.

Field	Type	Default	Description
`d_model`	int	—	Residual-stream / embedding width
`n_layers`	int	—	Number of MinMaxLayers
`d_state`	int	—	Hidden-state dimension of each neuron
`norm`	str	`'layernorm'`	Pre-norm inside each layer: `'layernorm'`, `'rmsnorm'`, `'none'`
`postlayers_norm`	str	`'layernorm'`	Norm applied after the final layer
`ffn_type`	str	`'gated'`	`'gated'` (ReGLU/SwiGLU) or `'basic'` (MLP)
`ffn_proj_factor`	float	`1.3`	FFN hidden-dim expansion factor
`ffn_act_fn`	str	`'relu'`	Activation: `'relu'`, `'gelu'`, `'swish'`, `'sigmoid'`, `'selu'`
`ffn_dropout`	float	`0.1`	Dropout inside the FFN
`ffn_init`	str	`'scaled'`	`'basic'` (PyTorch default) or `'scaled'` (`small_init` + `wang_init`)
`output_gate`	bool	`True`	Gate the neuron output by a learned projection of the input
`train_init`	bool	`False`	Make the initial hidden state $\mathbf{x}_0$ a learned parameter
`neuron_dropout`	float	`0.0`	Dropout on the neuron input
`s_r_init`	str	`'small_init'`	Init for W_s, W_r: `'small_init'`, `'kaiming'`, `'asymmetric'`
`conv_type`	str	`'basic'`	`'gated'` (scalar gate) or `'basic'` (linear mix)
`conv_init_val`	float	`0.0`	Initial gate logit for GatedConv
`prelayers_dropout`	float	`0.0`	FFN dropout override for the first layer only
`use_postlayers_ffn`	bool	`False`	Add an FFN after all layers

Table 3: Description of the `MinMaxRNCLMConfig` dataclass.

Field	Type	Default	Description
`backbone`	`MinMaxRNCConfig`	—	Backbone configuration
`head_dropout`	float	`0.0`	Dropout before the LM head
`tie_weights`	bool	`True`	Share embedding and LM-head weights
`output_gate`	bool	`True`	Gate each neuron output by $\mathrm{sigmoid}(W_g\mathbf{u})$; overrides `backbone.output_gate`
`conv_type`	str	`'basic'`	Short-range conv variant; overrides `backbone.conv_type`
`ffn_dropout`	float	`0.1`	FFN dropout for all layers except the first (which uses `backbone.prelayers_dropout`); overrides `backbone.ffn_dropout`

P Details of the Experimental Evaluation on Group-free Benchmarks

This appendix provides further details of our empirical evaluation on synthetic benchmarks.

See the extended version [64] for details on DeltaNet and "Assessment of non-vanishing gradient".

Table of Contents

P.1 Benchmarks 256
P.2 Models 259
P.3 Trainers 261
P.4 Training Protocol 262
P.5 Evaluation Protocol 263
P.6 Overview of the Included Experiments 263
P.7 Results: Accuracy 263
P.8 Results: Throughput 267

P.1 Benchmarks

We consider three token-based synthetic benchmark tasks: `Latching`(n), `Sequences`(n), and `InductionHeads`(n). Each task is parametrised along a difficulty parameter n, and it consists in predicting output tokens given a sequence of input tokens.

The input sequences to the models are obtained by encoding the finite vocabulary $\mathcal{V} = \{v_1, \dots, v_m\}$ of a task into integers $[1..m]$ via a bijection $\mathrm{enc} : \mathcal{V} \to [1..m]$. Models are required to return logits over the vocabulary $\mathcal{V}$, and the predicted token is the one having maximum logit (with ties broken arbitrarily).

Next we describe the benchmark tasks. For each task, we first define a general parametrisation, and then we define the parametrisation in the difficulty parameter n by defining some parameters as functions of n and fixing the other parameters.

Latching. The `Latching` task is parametrised by the total number $V \in \mathbb{N}^{>0}$ of tokens, and the number $M \in \mathbb{N}^{>0}$ of tokens that can appear in the first position (with $V - M \geq 1$).

The vocabulary is the disjount union $\mathcal{V} = \mathcal{V}_{\mathrm{mem}} \cup \mathcal{V}_{\mathrm{other}}$ with

$$\mathcal{V}_{\mathrm{mem}} = \{mem_i \mid i \in [1..M]\}, \qquad \mathcal{V}_{\mathrm{other}} = \{other_i \mid i \in [1..V - M]\}.$$

On input $s = \sigma_1 \cdots \sigma_T$, the expected output sequence is $\gamma_1 \cdots \gamma_T$ where $\gamma_t = \sigma_1$ for every $t \in [1..T]$.

To define datasets, we introduce parameters specifying the minimum and maximum sequence length: $T_{\min}, T_{\max} \in \mathbb{N}$ satisfying $T_{\min} \leq T_{\max}$. Then, an input sequence $\mathcal{U}_1 \cdots \mathcal{U}_{\mathcal{T}}$ is generated according to the following distributions:

$$\mathcal{T} \sim \mathrm{Unif}([T_{\min}..T_{\max}]), \qquad \mathcal{U}_1 \sim \mathrm{Unif}(\mathcal{V}_{\mathrm{mem}}), \qquad \mathcal{U}_t \sim \mathrm{Unif}(\mathcal{V}_{\mathrm{other}}) \quad \forall t \in [2..\mathcal{T}].$$

For the experiments, we adopt a single-number parametrisation `Latching`(n) where increasing the value of $n \in \mathbb{N}^{>0}$ increases the difficulty of the task as it is the number of candidate tokens to memorise. Table 4 reports the details of the parameterisation in n, and also the values of the other parameters (derived or fixed) for the values of n that we consider in the experiments.

Sequences. The `Sequences` task is parametrised by the number $N \in \mathbb{N}^{>0}$ of distinct sequences to match, the length $L \in \mathbb{N}^{>0}$ of the sequences to match, the number $M \in \mathbb{N}^{>0}$ of distinct tokens for each event type, and the number $I \in \mathbb{N}^{>0}$ of irrelevant tokens.

Table 4: Details of Latching(n).

Training

	Parameter values					
Parametrisation	V	M	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
Latching(n)	$5n$	n	256	512	$5n$	20k
Latching(4)	20	4	256	512	20	20k
Latching(8)	40	8	256	512	40	20k
Latching(16)	80	16	256	512	80	20k
Latching(32)	160	32	256	512	160	20k
Latching(64)	320	64	256	512	320	20k
Latching(128)	640	128	256	512	640	20k
Latching(256)	1280	256	256	512	1280	20k
Latching(512)	2560	512	256	512	2560	20k

Validation

	Parameter values					
Parametrisation	V	M	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
Latching(n)	$5n$	n	1024	2048	$5n$	1k
Latching(4)	20	4	1024	2048	20	1k
Latching(8)	40	8	1024	2048	40	1k
Latching(16)	80	16	1024	2048	80	1k
Latching(32)	160	32	1024	2048	160	1k
Latching(64)	320	64	1024	2048	320	1k
Latching(128)	640	128	1024	2048	640	1k
Latching(256)	1280	256	1024	2048	1280	1k
Latching(512)	2560	512	1024	2048	2560	1k

Evaluation

	Parameter values					
Parametrisation	V	M	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
Latching(n)	$5n$	n	2^{10}	2^{10}	$5n$	1k
Latching(4)	20	4	2^{10}	2^{10}	20	1k
Latching(8)	40	8	2^{10}	2^{10}	40	1k
Latching(16)	80	16	2^{10}	2^{10}	80	1k
Latching(32)	160	32	2^{10}	2^{10}	160	1k
Latching(64)	320	64	2^{10}	2^{10}	320	1k
Latching(128)	640	128	2^{10}	2^{10}	640	1k
Latching(256)	1280	256	2^{10}	2^{10}	1280	1k
Latching(512)	2560	512	2^{10}	2^{10}	2560	1k

The vocabulary consists of input tokens from the disjoint sets $\mathcal{V}_{\text{set}}, \mathcal{V}_{\text{reset}}, \mathcal{V}_{\text{others}}$ defined as follows:

$$\mathcal{V}_{\text{set},n,\ell,m} = \{set_{n,\ell,m} \mid m \in [1..M]\}, \qquad \mathcal{V}_{\text{set},n} = \bigcup_{\ell=1}^{L} \mathcal{V}_{\text{set},n,\ell}, \qquad \mathcal{V}_{\text{set}} = \bigcup_{n=1}^{N} \mathcal{V}_{\text{set},n};$$

$$\mathcal{V}_{\text{reset},n} = \{reset_{n,m} \mid m \in [1..M]\}, \qquad \mathcal{V}_{\text{reset}} = \bigcup_{n=1}^{N} \mathcal{V}_{\text{reset},n};$$

$$\mathcal{V}_{\text{other}} = \{other_i \mid i \in [1..I]\},$$

and it also contains output tokens representing the subsets of $[1..N]$.

For an input sequence $s = \sigma_1 \cdots \sigma_T$, and for $n \in [1..N]$ and $t \in [1..T]$, we say that *the n-th sequence is matched in s at time t* if there are $t_1 < \cdots < t_L \leq t$ such that $\sigma_{t_\ell} \in \mathcal{V}_{\text{set},n,\ell}$ and there is no t' with $t_1 < t' \leq t$ such that $\sigma_{t'} \in \mathcal{V}_{\text{reset},n}$. Then, the expected output sequence $\gamma_1 \cdots \gamma_T$ is the one where γ_t is the token representing the set of indices n such that the n-th sequence is matched in s at time t.

To define datasets, we introduce the following parameters: $p_{\text{set}}, p_{\text{reset}}, p_{\text{other}} \in [0, 1]$ satisfying $p_{\text{set}} + p_{\text{reset}} + p_{\text{other}} = 1$, and $T_{\min}, T_{\max} \in \mathbb{N}$ satisfying $T_{\min} \leq T_{\max}$. Then, an input sequence $\mathcal{U}_1 \cdots \mathcal{U}_\mathcal{T}$ is generated by first drawing $\mathcal{T} \sim \text{Unif}([T_{\min}..T_{\max}])$ and then drawing tokens according to the following probabilities:

$$\text{P}(\mathcal{U}_t = set_{n,\ell,m}) = \frac{p_{\text{set}}}{|\mathcal{V}_{\text{set}}|}, \qquad \text{P}(\mathcal{U}_t = reset_{n,\ell,m}) = \frac{p_{\text{reset}}}{|\mathcal{V}_{\text{reset}}|}, \qquad \text{P}(\mathcal{U}_t = other_i) = \frac{p_{\text{other}}}{|\mathcal{V}_{\text{other}}|}.$$

For the experiments, we adopt a single-number parametrisation `Sequences`(n) where increasing the value of $n \in \mathbb{N}^{>0}$ increases the difficulty of the task as it corresponds to the length of the sequences to match. Table 5 reports the details of the parameterisation in n, and also the values of the other parameters (derived or fixed) for the values of n that we consider in the experiments.

Table 5: Details of `Sequences`(n).

Training

	Parameter values									
Parametrisation	N	L	M	I	p_{set}	p_{reset}	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
`Sequences`(n)	2	n	2	10	0.5	0.02	256	512	$4n$+19	20k
`Sequences`(2)	2	2	2	10	0.5	0.02	256	512	27	20k
`Sequences`(3)	2	3	2	10	0.5	0.02	256	512	31	20k
`Sequences`(4)	2	4	2	10	0.5	0.02	256	512	35	20k
`Sequences`(8)	2	8	2	10	0.5	0.02	256	512	51	20k
`Sequences`(16)	2	16	2	10	0.5	0.02	256	512	83	20k

Validation

	Parameter values									
Parametrisation	N	L	M	I	p_{set}	p_{reset}	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
`Sequences`(n)	2	n	2	10	0.3	0.002	1024	2048	$4n$+19	1k
`Sequences`(2)	2	2	2	10	0.3	0.002	1024	2048	27	1k
`Sequences`(3)	2	3	2	10	0.3	0.002	1024	2048	31	1k
`Sequences`(4)	2	4	2	10	0.3	0.002	1024	2048	35	1k
`Sequences`(8)	2	8	2	10	0.3	0.002	1024	2048	51	1k
`Sequences`(16)	2	16	2	10	0.3	0.002	1024	2048	83	1k

Evaluation

	Parameter values									
Parametrisation	N	L	M	I	p_{set}	p_{reset}	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
`Sequences`(n)	2	n	2	10	10^{-5}	10^{-6}	2^{10}	2^{10}	$4n$+19	1k
`Sequences`(2)	2	2	2	10	10^{-5}	10^{-6}	2^{10}	2^{10}	27	1k
`Sequences`(3)	2	3	2	10	10^{-5}	10^{-6}	2^{10}	2^{10}	31	1k
`Sequences`(4)	2	4	2	10	10^{-5}	10^{-6}	2^{10}	2^{10}	35	1k
`Sequences`(8)	2	8	2	10	10^{-5}	10^{-6}	2^{10}	2^{10}	51	1k
`Sequences`(16)	2	16	2	10	10^{-5}	10^{-6}	2^{10}	2^{10}	83	1k

Induction Heads. The `InductionHeads` task is parametrised by the number $M \in \mathbb{N}^{>0}$ of candidate tokens to recall.

The vocabulary is $\mathcal{V} = \mathcal{V}_{\text{mem}} \cup \{recall, pad\}$ where $\mathcal{V}_{\text{mem}}$ is a set of M tokens, $recall$ is the recall token (not included in $\mathcal{V}$), and pad is the padding token (not included in $\mathcal{V}$).

Input sequences are of the form

$$s = \sigma_1, \ldots, \sigma_{t_0-1}, \mathit{recall}, \sigma_{t_0+1}, \ldots, \sigma_T, \mathit{recall} \qquad \text{where} \quad \sigma_t \in \mathcal{V}_{\mathrm{mem}},$$

and the expected output is σ_{t_0+1}.

To define datasets, we introduce the parameters $T^0_{\min}, T^0_{\max}, T_{\min}, T_{\max} \in \mathbb{N}$ satisfying $T^0_{\min} \leq T^0_{\max} \leq T_{\min} \leq T_{\max}$. Then, an input sequence $\mathcal{U}_1 \cdots \mathcal{U}_{\mathcal{T}}$ is generated by first drawing

$$\mathcal{T}_0 \sim \mathrm{Unif}([T^0_{\min}..T^0_{\max}]), \qquad \mathcal{T} \sim \mathrm{Unif}([T_{\min}..T_{\max}]),$$

then setting $\mathcal{U}_{\mathcal{T}_0} = \mathcal{U}_{\mathcal{T}} = \mathit{recall}$, and finally drawing tokens for the other positions:

$$\mathcal{U}_t \sim \mathrm{Unif}(\mathcal{V}) \qquad \forall t \in [1..\mathcal{T}_0) \cup (\mathcal{T}_0..\mathcal{T}).$$

For the experiments, we adopt a single-number parametrisation `InductionHeads`(n) where increasing the value of $n \in \mathbb{N}^{>0}$ increases the difficulty of the task as it is the number of candidate tokens to recall. Table 6 reports the details of the parameterisation in n, and also the values of the other parameters (derived or fixed) for the values of n that we consider in the experiments.

Table 6: Details of `InductionHeads`(n).

Training

	Parameter values						
Parametrisation	M	$T^0_{\min}$	$T^0_{\max}$	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
`InductionHeads`(n)	n	0	30	35	512	n+2	20k
`InductionHeads`(16)	16	0	30	35	512	18	20k
`InductionHeads`(30)	30	0	30	35	512	32	20k

Validation

	Parameter values						
Parametrisation	M	$T^0_{\min}$	$T^0_{\max}$	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
`InductionHeads`(n)	n	0	50	1024	2048	n+2	1k
`InductionHeads`(16)	16	0	50	1024	2048	18	1k
`InductionHeads`(30)	30	0	50	1024	2048	32	1k

Evaluation

	Parameter values						
Parametrisation	M	$T^0_{\min}$	$T^0_{\max}$	$T_{\min}$	$T_{\max}$	$\lvert\mathcal{V}\rvert$	Size
`InductionHeads`(n)	n	0	50	2^{10}	2^{10}	n+2	1k
`InductionHeads`(16)	16	0	50	2^{10}	2^{10}	18	1k
`InductionHeads`(30)	30	0	50	2^{10}	2^{10}	32	1k

P.2 Models

The experiments include the following models.

MinMax We use the implementation described in Appendix O. When we specify the flag OG we mean `output_gate=True`, and otherwise `output_gate=False`. When we specify the flag GC we mean `conv_type='gated'`, and otherwise `conv_type='basic'`. All other parameters are left to their default value.

Mamba Introduced in [13]. We use the official PyTorch implementation [69] version `2.2.4`. We scale its size by varying `d_model` and the number L of stacked Mamba blocks, while setting all the other parameters to the values specified in [13]: `d_state` = 16, `d_conv` = 4, `expand` = 2, no dropout.

Table 7: Model configurations used in the experiments, along with dimensions and parameter count. *Legend:* V is the vocabulary size, and L is the number of layers.

		Dimensions			
Model	**Config**	d_{model}	d_{state}	d_{conv}	**Trainable Parameters**
MinMax	S	90	40	90	$60\text{k} \cdot L + 90 \cdot V + 180$
MinMax	S, OG	90	40	90	$64\text{k} \cdot L + 90 \cdot V + 180$
MinMax	S, GC	90	40	90	$44\text{k} \cdot L + 90 \cdot V + 180$
MinMax	S, OG, GC	90	40	90	$47\text{k} \cdot L + 90 \cdot V + 180$
MinMax	M	90	90	90	$74\text{k} \cdot L + 90 \cdot V + 180$
MinMax	M, OG	90	90	90	$82\text{k} \cdot L + 90 \cdot V + 180$
MinMax	M, GC	90	90	90	$57\text{k} \cdot L + 90 \cdot V + 180$
MinMax	M, OG, GC	90	90	90	$66\text{k} \cdot L + 90 \cdot V + 180$
Mamba	S	95	3040	760	$67\text{k} \cdot L + 95 \cdot V$
Mamba	M	100	3200	800	$74\text{k} \cdot L + 100 \cdot V$
sLSTM	S	96	384	288	$56\text{k} \cdot L + 192 \cdot V + 96$
sLSTM	M	100	400	300	$79\text{k} \cdot L + 200 \cdot V + 100$
mLSTM	S	96	9412	576	$64\text{k} \cdot L + 192 \cdot V + 96$
mLSTM	M	97	16644	768	$86\text{k} \cdot L + 194 \cdot V + 97$
xLSTM-L	S	72	688	0	$60\text{k} \cdot L + 90 \cdot V + 360$

Table 8: Number of trainable parameters for each model on each benchmark. *Legend:* V is the vocabulary size; L is the number of layers. The range reported for MinMax aggregates the counts for different configurations obtained via the optional flags OG and GC.

`Latching`(n)

n	V	Config	L	MinMax	Mamba	xLSTM-L	mLSTM	sLSTM
4	20	S	2	90–130k	136k	129k	131k	117k
8	40	S	2	91–131k	138k	132k	135k	121k
16	80	S	2	95–135k	142k	138k	143k	128k
32	160	S	2	102–142k	149k	149k	158k	144k
64	320	S	2	117–156k	165k	172k	189k	174k
128	640	S	2	145–185k	195k	218k	250k	236k
256	1280	S	2	203–243k	256k	310k	373k	359k
512	2560	S	2	318–358k	377k	495k	619k	605k

`Sequences`(n)

n	V	Config	L	MinMax	Mamba	xLSTM-L	mLSTM	sLSTM
2	27	S	2	90–130k	137k	130k	133k	118k
3	31	S	3	134–194k	204k	194k	197k	175k
4	35	S	4	178–258k	272k	257k	261k	233k
8	51	S	8	224–323k	340k	322k	328k	292k
16	83	S	16	270–389k	410k	390k	398k	355k

`InductionHeads`(n)

n	V	Config	L	MinMax	Mamba	xLSTM-L	mLSTM	sLSTM
16	18	S	2	89–129k	136k	129k	131k	116k
30	32	S	2	91–130k	137k	131k	134k	119k
16	18	M	2	117–165k	150k	—	175k	161k
30	32	M	2	118–167k	151k	—	178k	164k

sLSTM By sLSTM we mean xLSTM consisting of sLSTM cells only [6]. We use the official PyTorch implementation [70] version `2.0.5`. We scale its size by varying `embedding_dim` and the number L of stacked cells, while setting all the other parameters to the values specified in [6]: `num_heads` = 4, `conv1d_kernel_size` = 4, `proj_factor` = 1.3, `act_fn` = GeLU, `dropout` = 0.1.

mLSTM By mLSTM we mean xLSTM consisting of mLSTM cells only [6]. We use the official PyTorch implementation [70] version `2.0.5`. We scale its size by varying `embedding_dim` and the number L of stacked cells, while setting all the other parameters to the values specified in [6]: `num_heads` = 4, `qkv_proj_block_size` = 4, `conv1d_kernel_size` = 4, `proj_factor` = 1.3, `act_fn` = GeLU, `dropout` = 0.1.

xLSTM-L Introduced in [32]. We use the official PyTorch implementation [70] version `2.0.5`. We scale its size by varying `embedding_dim` and the number L of stacked cells, while setting `num_heads` = 4 and leaving the other parameters to their default values.

Table 7 reports the model configurations we consider in the experiments. For each model we can have a smaller variant (S) and a larger variant (M). The table reports the formula to compute the number of trainable parameters as a function of the number of layers L and vocabulary size V, and it also reports the following dimensions for each model:

- d_{model} which is `d_model` for MinMax and Mamba, and it is `embedding_dim` for sLSTM, mLSTM, xLSTM-L;
- d_{state} which is the total number of scalar values in the state of a single layer;
- d_{conv} which is the dimension of the convolution buffer for a single layer.

The above dimensions allow for a closer comparison of the models rather than simply comparing the total number of trainable parameters. Interestingly, the table makes it evident that the state dimension d_{state} of MinMax RNCs is significantly smaller than the one of the other models, even though all models have comparable d_{model} and parameter count. We have first chosen $d_{\text{model}} = 90$ for MinMax RNC as an educated guess for a reasonable value, and then we have chosen d_{model} for the other models in order to obtain models having a comparable number of a trainable parameters.

Table 8 reports parameter counts for the model configurations considered in the experiments.

P.3 Trainers

We introduce the notion of a *trainer* to group the choice of the optimiser, learning rate schedule, and set of batch sizes. We use a dedicated trainer for each family of models. They are defined in Table 9, and further described and discussed below.

Table 9: Trainers.

	Optimiser				Learning Rate		
Trainer	Type	λ	β_1	β_2	Peak	Schedule	**Batch sizes**
`trainer_minmax`	Adam	10^{-4}	0.9	0.999	10^{-3}	Step (γ=0.5, step=5)	64
`trainer_minmax_bs8`	Adam	10^{-4}	0.9	0.999	10^{-3}	Step (γ=0.5, step=5)	8
`trainer_xlstm`	AdamW	0.1	0.9	0.99	10^{-3}	Linear warmup + cosine	64, 128
`trainer_xlstm_bs8`	AdamW	0.1	0.9	0.99	10^{-3}	Linear warmup + cosine	8
`trainer_mamba`	Adam	no decay	0.9	0.999	10^{-3}	None	64
`trainer_mamba_bs8`	Adam	no decay	0.9	0.999	10^{-3}	None	8

MinMax. The training configuration for MinMax is Adam optimiser with weight decay $\lambda = 10^{-4}$, learning rate 10^{-3}, and a step learning-rate schedule that halves the learning rate every 5 epochs ($\gamma = 0.5$); batch size 64 for the default trainer, and batch size 8 for the dedicated trainer. The rationale is to provide a rather standard and basic training setup.

xLSTM. The training configuration for sLSTM, mLSTM, and xLSTM-L is AdamW optimiser with $(\beta_1, \beta_2) = (0.9, 0.99)$, weight decay 0.1, $\varepsilon = 10^{-5}$, gradient clipping at 1.0; learning rate 10^{-3}

with a linear warmup over the first 10% of total training steps followed by cosine annealing; batch size ranges over $\{64, 128\}$ for the default trainer, and we include a dedicate trainer using batch size 8. The configuration is in line with the one used in [6] for the experiments on synthetic benchmarks based on formal languages.

Mamba. The training configuration for Mamba is Adam optimiser without weight decay, learning rate 10^{-3}, no learning-rate schedule, gradient clipping at 1.0. For batch size 8, it is the configuration used in [13] for the InductionHeads benchmark. We also include the batch-size-64 variant for a closer comparison with the other models—also noting that this configuration is in fact used in [13] for benchmarks other than InductionHeads (e.g., Selective Copying).

P.4 Training Protocol

Our training protocol is summarised in Table 10.

The loss function is *cross-entropy* over all non-masked token positions. The metric is *token accuracy*: the number of correctly predicted tokens over the total number of tokens to predict.

Each model-trainer pair defines a separate experiment. For each batch size specified by the trainer, we run independent trainings for different random seeds until we reach 99% confidence that no training run will produce a model that has validation accuracy 0.1% higher than the one of the best model found so far—trying at least 3 seeds to avoid poor statistical estimates, and no more than 10 seeds to limit the amount of compute.

We estimate 99% confidence intervals (CIs) for accuracies using the *nonparametric percentile bootstrap* over seeds with 10k resamples.

We use independently-generated training and validation datasets (per seed), having the characteristics reported in Tables 4, 5, 6. In particular, training datasets consist of 20k examples, and validation datasets consist of 1k examples.

We use epoch-based training, where each epoch amounts to a full pass over the training dataset. Models are trained for a maximum of *30 epochs with early stopping*. Specifically, we employ patience-based early stopping on the validation loss. The criterion monitors a sliding window of 3 epochs and triggers if no improvement greater than $\delta = 10^{-5}$ is observed for 5 consecutive windows (patience = 5).

We set a *time limit* of 4 hours for each model-trainer pair, in order to contain the total computation time to carry out the experiments.

The *best model* resulting from a full training run is the one that has highest validation accuracy across all models (for all batch sizes and considered seeds); for a training run that times out, it is the best model obtained when the run times out.

Although the training protocol is independent of the specific datasets considered, we recall size and sequence length of the datasets we consider in our experiments (as described in Tables 4, 5, 6):

- training datasets consist of 20k labelled sequences of length $[256..512]$ for `Latching` and `Sequences`, and of length $[35..512]$ for `InductionHeads`;
- all validation datasets consist of 1k labelled sequences of length $[1024..2048]$.

Table 10: Overview of the training protocol.

				CI			
Loss	**Metric**	**Epochs**	**Early stop**	**Confidence**	**Margin**	**Method**	**Time limit**
cross-entropy	token accuracy	30	patience-based	99%	0.1%	bootstrap	4h

P.5 Evaluation Protocol

After training, the best models are evaluated on long sequences length 2^{10} ($\approx$ 1M). We set a time limit of 20 hours to evaluate a single model. We measure accuracy at each non-masked sequence steps, also computing the average across non-masked steps and the minimum accuracy at any non-masked step. We recall that on `Latching` and `Sequences` no step is masked, whereas on `InductionHeads` all steps are masked except the last one—hence accuracy is the one at the last step. We repeat the evaluation for different random seeds (with an independently-generated dataset for each seed) until the 99%-CIs for the step-average accuracy and step-minimum accuracy both satisfy the following (with CIs computed as in training):

- if the mean is in the range 0–50%, then the error margin is below 20%;
- if the mean is in the range 50%–80%, then the error margin is below 10%;
- if the mean is in the range 80%–90%, then the error margin is below 5%;
- if the mean is in the range 90%–100%, then the error margin is below 1%.

Although the evaluation protocol is independent of the specific datasets considered, we recall that the details of the datasets for our experiments are described in Tables 4, 5, 6. In particular, each dataset consists of 1k labelled sequences of length 2^{10} ($\approx$ 1M).

P.6 Overview of the Included Experiments

Table 11 summarises our experiments on the synthetic benchmarks. For every benchmark, value of n, and included model configuration, we carry out training and evaluation for the listed model-trainer pairs; also including the trainer variants using batch size 8 when the table specifies "+ `bs8`".

Table 11: Overview of the experiments.

Benchmark	n	**Configurations**	**Model**	**Trainer**
`Latching`(2^n)	$\{2, 3, \ldots, 9\}$	small (S)	MinMax	`trainer_minmax`
			Mamba	`trainer_mamba` + `bs8`
			sLSTM	`trainer_xlstm`
			mLSTM	`trainer_xlstm`
			xLSTM-L	`trainer_xlstm`
`Sequences`(n)	$\{2, 3, 4, 8, 16\}$	small (S)	MinMax	`trainer_minmax`
			Mamba	`trainer_mamba`
			sLSTM	`trainer_xlstm`
			mLSTM	`trainer_xlstm`
			xLSTM-L	`trainer_xlstm`
`InductionHeads`(n)	$\{16, 30\}$	small (S), medium (M)	MinMax	`trainer_minmax` + `bs8`
			Mamba	`trainer_mamba` + `bs8`
			sLSTM	`trainer_xlstm` + `bs8`
			mLSTM	`trainer_xlstm` + `bs8`
			xLSTM-L	`trainer_xlstm` + `bs8`

P.7 Results: Accuracy

Table 12 summarises the results of our evaluation. For each benchmark instance, the table reports the best evaluation accuracy of each model across the considered configurations—i.e., model size and trainer. For `Latching`(n) and `Sequences`(n) the reported accuracy is the step-average accuracy; and for `InductionHeads`(n) is the last-step accuracy.

Tables 13, 14, and 15 provide additional details for the evaluation on `Latching`(n), `Sequences`(n), and `InductionHeads`(n), respectively—they report the train, validation, and evaluation accuracy for each model-trainer pair.

Table 12: Summary of the evaluation: *mean* ± *error margin* of 99% CIs for evaluation accuracy (%). *Legend:* ✓ denotes perfect accuracy on all seeds.

	Accuracy (%)				
Benchmark	MinMax	xLSTM-L	mLSTM	sLSTM	Mamba
`Latching`(4)	✓	25.4 ± 0.5	25.5 ± 1.0	✓	52.3 ± 2.3
`Latching`(8)	✓	12.5 ± 0.8	13.5 ± 0.8	62.8 ± 1.4	70.5 ± 0.6
`Latching`(16)	✓	6.1 ± 0.4	7.8 ± 0.3	40.4 ± 0.2	42.6 ± 1.5
`Latching`(32)	✓	4.4 ± 0.5	3.7 ± 0.4	4.6 ± 0.3	12.9 ± 0.3
`Latching`(64)	✓	3.3 ± 0.3	2.0 ± 0.3	5.1 ± 0.2	99.9
`Latching`(128)	✓	5.9 ± 0.3	2.2 ± 0.1	3.4 ± 0.2	90.6 ± 1.0
`Latching`(256)	✓	7.5 ± 0.5	1.8 ± 0.2	2.7 ± 0.4	64.1 ± 1.3
`Latching`(512)	✓	0.1 ± 0.1	2.4 ± 0.2	6.6 ± 0.2	65.2 ± 1.7
`Sequences`(2)	✓	47.0 ± 1.1	47.0 ± 1.2	47.5 ± 1.0	47.5 ± 1.0
`Sequences`(3)	✓	63.1 ± 0.9	47.1 ± 0.2	63.1 ± 0.9	63.1 ± 0.9
`Sequences`(4)	✓	74.1 ± 1.0	74.1 ± 1.0	74.1 ± 1.0	74.1 ± 1.0
`Sequences`(8)	✓	90.5 ± 0.5	90.5 ± 0.5	90.5 ± 0.5	90.5 ± 0.5
`Sequences`(16)	✓	97.0 ± 0.3	97.0 ± 0.3	96.8 ± 0.2	97.0 ± 0.3
`InductionHeads`(16)	✓	6.2 ± 0.7	6.5 ± 1.0	6.2 ± 0.2	✓
`InductionHeads`(30)	✓	3.1 ± 0.9	3.0 ± 0.6	3.0 ± 0.6	3.5 ± 0.2

Table 13: Results for `Latching`(n): training accuracy of the best model, and *mean* $\pm$ *error margin* of 99% CIs for validation and evaluation accuracy (%). *Legend:* ✓ perfect accuracy on all seeds; asterisk * denotes time-out at training time.

Accuracy (%) *(Part I)*

		`Latching`(4)			`Latching`(8)			`Latching`(16)		
Model	**Config**	Train	Validation	Evaluation	Train	Validation	Evaluation	Train	Validation	Evaluation
MinMax	small	✓	✓	✓	✓	✓	✓	✓	✓	✓
xLSTM-L	small	✓	✓	25.4 ± 0.5	✓	✓	12.5 ± 0.8	✓	✓	6.1 ± 0.4
xLSTM	small	✓	✓	25.5 ± 1.0	✓	✓	13.5 ± 0.8	✓	✓	7.8 ± 0.3
sLSTM	small	✓*	✓*	✓	✓*	✓*	62.8 ± 1.4	✓*	✓*	40.4 ± 0.2
Mamba	small	✓	✓	52.3 ± 2.3	✓	✓	38.8 ± 1.5	✓	✓	42.6 ± 1.5
Mamba	small,bs8	✓	✓	27.7 ± 1.2	✓	✓	70.5 ± 0.6	✓	✓	36.0 ± 1.3

Accuracy (%) *(Part II)*

		`Latching`(32)			`Latching`(64)			`Latching`(128)		
Model	**Config**	Train	Validation	Evaluation	Train	Validation	Evaluation	Train	Validation	Evaluation
MinMax	small	✓	✓	✓	✓	✓	✓	✓	✓	✓
xLSTM-L	small	99.9	✓	4.4 ± 0.5	99.9	✓	3.3 ± 0.3	99.9	✓	5.9 ± 0.3
xLSTM	small	✓	✓	3.7 ± 0.4	✓	✓	2.0 ± 0.3	✓	✓	2.2 ± 0.1
sLSTM	small	✓*	✓*	4.6 ± 0.3	✓*	✓*	5.1 ± 0.2	✓*	✓*	3.4 ± 0.2
Mamba	small	✓	✓	12.9 ± 0.3	✓	✓	14.4 ± 0.9	✓	✓	87.3 ± 1.6
Mamba	small,bs8	✓	✓	13.0 ± 1.3	✓	✓	99.9	✓	✓	90.6 ± 1.0

Accuracy (%) *(Part III)*

		`Latching`(256)			`Latching`(512)		
Model	**Config**	Train	Validation	Evaluation	Train	Validation	Evaluation
MinMax	small	✓	✓	✓	✓	✓	✓
xLSTM-L	small	99.9	99.8 ± 0.1	7.5 ± 0.5	0.2	0.4 ± 0.1	0.1 ± 0.1
xLSTM	small	✓	✓	1.8 ± 0.2	✓	✓	2.4 ± 0.2
sLSTM	small	✓*	✓*	2.7 ± 0.4	✓*	✓*	6.6 ± 0.2
Mamba	small	✓	✓	64.1 ± 1.3	✓	✓	6.5 ± 0.5
Mamba	small,bs8	✓	✓	31.5 ± 1.5	✓	✓	65.2 ± 1.7

Table 14: Results for `Sequences`(n): training accuracy of the best model, and *mean* $\pm$ *error margin* of 99% CIs for validation and evaluation accuracy (%). *Legend:* ✓ perfect accuracy on all seeds; asterisk * denotes time-out at training time.

Accuracy (%) *(Part I)*

		`Sequences`(2)			`Sequences`(3)			`Sequences`(4)		
Model	**Config**	Train	Validation	Evaluation	Train	Validation	Evaluation	Train	Validation	Evaluation
MinMax	small	✓	✓	✓	✓	✓	✓	✓	✓	✓
xLSTM-L	small	✓	✓	47.0±1.1	✓	✓	63.1±0.9	✓	✓	74.1±1.0
xLSTM	small	✓*	✓*	47.0±1.2	✓*	✓*	47.1±0.2	✓*	✓*	74.1±1.0
sLSTM	small	99.9	99.9	47.5±1.0	✓	99.9	63.1±0.9	✓	99.9	74.1±1.0
Mamba	small	✓	✓	47.5±1.0	✓	✓	63.1±0.9	✓	✓	74.1±1.0

Accuracy (%) *(Part II)*

		`Sequences`(8)			`Sequences`(16)		
Model	**Config**	Train	Validation	Evaluation	Train	Validation	Evaluation
MinMax	small	✓	✓	✓	✓	✓	✓
xLSTM-L	small	99.9	✓	90.5±0.5	✓	✓	97.0±0.3
xLSTM	small	✓*	✓*	90.5±0.5	✓*	99.9*	97.0±0.3
sLSTM	small	99.9	99.9	90.5±0.5	99.9	99.9	96.8±0.2
Mamba	small	✓	✓	90.5±0.5	✓	✓	97.0±0.3

Table 15: Results for `InductionHeads`(n): training accuracy of the best model, and *mean* $\pm$ *error margin* of 99% CIs for validation and evaluation accuracy (%). *Legend:* L2/L3 denotes 2/3 layers; ✓ perfect accuracy on all seeds; asterisk * denotes time-out at training time; — not executed (noting low validation accuracy).

Accuracy (%)

		`InductionHeads`(16)			`InductionHeads`(30)		
Model	**Config**	Train	Validation	Evaluation	Train	Validation	Evaluation
MinMax	L2,small	✓	✓	✓	✓	✓	✓
Mamba	L2,small	8.5	7.5	5.5±1.0	5.1	4.1	3.5±0.4
xLSTM	L2,small	✓	✓	6.5±1.0	✓	✓	3.0±0.6
sLSTM	L2,small	✓*	87.2*	6.2±0.2*	✓*	88.3	3.0±0.6
xLSTM-L	L2,small	7.2	8.3	6.2±0.7	3.5	4.7	3.1±0.9
MinMax	L2,small,bs8	✓	✓	✓	✓	✓	✓
Mamba	L2,small,bs8	✓	✓	✓	4.7	5.2	3.5±0.2
xLSTM	L2,small,bs8	✓	✓	6.1±1.3	✓	✓	3.0±0.2
sLSTM	L2,small,bs8	99.9*	65.4*	5.9±0.8	99.9*	80.8*	3.7±1.4
xLSTM-L	L2,small,bs8	7.1	6.6	6.7±1.1	4.1	4.6	3.0±0.2
MinMax	L2,medium	✓	✓	✓	✓	✓	✓
Mamba	L2,medium	8.0	8.0	6.2±1.0	3.8	4.0	—
xLSTM	L2,medium	✓	✓	6.0±1.0	✓	✓	3.4±0.2
sLSTM	L2,medium	✓*	99.2*	5.7±1.0	✓*	83.3*	3.7±0.5
MinMax	L3,small	✓	✓	✓	✓	✓	✓
Mamba	L3,small	7.3	7.0	6.0±0.7	4.5	4.3	—
sLSTM	L3,small	99.0*	61.3*	6.5±0.9	✓*	79.1*	3.7±0.5
xLSTM	L3,small	✓	✓	6.5±1.0	✓	✓	3.4±0.6
xLSTM-L	L3,small	6.6	8.1	5.8±0.2	4.0	4.9	—

P.8 Results: Throughput

We complement the accuracy evaluation with an empirical assessment of computational efficiency. We measure *training throughput* and *evaluation throughput* in processed tokens per second, using the same model configurations and single A100 GPU employed in the experiments above. For each model, we report throughput averaged across the considered benchmark instances; the uncertainty is the standard error (SE), expressed as a percentage of the mean.

For MinMax, in addition to the implementation used for the experiments above, we consider an optimised implementation employing `torch.compile` and PyTorch automatic mixed precision (AMP). We rerun the MinMax experiments with these optimisations and obtain the same accuracy on every benchmark. Thus, the resulting throughput improvement does not come at the cost of predictive performance.

Table 16 reports the results. The ratios are relative to MinMax with compilation and AMP.

Table 16: Training and evaluation throughput averaged across the considered benchmark instances. Ratios are relative to MinMax with compilation and AMP. SE is reported as a percentage of the mean.

	Training		**Evaluation**	
Model	**tok/s**	**Ratio**	**tok/s**	**Ratio**
MinMax (compiled+AMP)	$1{,}136{,}600 \pm 7\%$	$1.00\times$	$10{,}853 \pm 9\%$	$1.00\times$
MinMax	$206{,}127 \pm 5\%$	$0.18\times$	$1{,}528 \pm 5\%$	$0.14\times$
Mamba	$724{,}930 \pm 4\%$	$0.64\times$	$595 \pm 1\%$	$0.05\times$
xLSTM-L	$1{,}606{,}865 \pm 2\%$	$1.41\times$	$9{,}210 \pm 5\%$	$0.85\times$
mLSTM	$206{,}524 \pm 2\%$	$0.18\times$	$249 \pm 3\%$	$0.02\times$
sLSTM	$9{,}323 \pm 7\%$	$0.01\times$	$207 \pm 3\%$	$0.02\times$
DeltaNet	$591{,}412 \pm 5\%$	$0.52\times$	$10{,}372 \pm 6\%$	$0.96\times$

Compilation and AMP increase MinMax throughput from 206k to 1.14M tokens/s during training, a $5.5\times$ speedup, and from 1.53k to 10.9k tokens/s during evaluation, a $7.1\times$ speedup. With these optimisations, MinMax has the second-highest training throughput among the considered models: xLSTM-L is $1.41\times$ faster, while MinMax is $1.57\times$ faster than Mamba and $1.92\times$ faster than DeltaNet. During evaluation, MinMax has the highest measured throughput, exceeding DeltaNet by $1.05\times$, xLSTM-L by $1.18\times$, and Mamba by $18.2\times$.

These measurements indicate that the favourable parallel complexity of MinMax RNCs is reflected in practical execution time. In particular, MinMax is competitive with the fastest considered parallel recurrent architectures in both training and evaluation, while the compiled+AMP variant retains the accuracy reported in the preceding experiments.

Q Details of Next-Token Prediction on OpenWebText with MinMax RNCs

This appendix provides further details on the next-token prediction experiment.

Overview. We train a MinMax RNC as a causal language model on the OpenWebText corpus using the GPT-2 tokeniser. The model has ~112M unique trainable parameters. The run completed 100k gradient steps in 14 days and 7 hours on a single A100 GPU.

Q.1 Setup

Details of the setup are provided in Table 17. The loss function is cross-entropy over the next token. The computed validation loss is a Monte Carlo estimate: it is obtained by averaging over 2,000 batches drawn randomly with replacement from the held-out validation set (each batch has 4 sequences of 1,024 tokens). Letting μ and σ be the mean and standard deviation of the validation loss, and letting $\hat{\mu}$ be the mean of $n = 2000$ i.i.d. batch losses, by the Central Limit Theorem we have $\hat{\mu} \approx \mathcal{N}(\mu, \sigma^2/n)$, and hence a 95% CI is $\hat{\mu} \pm 1.96\,\sigma/\sqrt{n}$. As we do not save individual batch losses, we estimate the standard error $\mathrm{SE} = \sigma/\sqrt{n}$ from the residual standard deviation of k consecutive losses after linear detrending.

Table 17: Setup.

Model

`minmaxrnc v0.1.0` [71]	
`d_model` = 728	
`d_state` = 1,456	
`n_layers` = 12	
`conv_type` = `gated`	
`gated_output` = `True`	
Total parameters: 112,467,448	

Training

Optimiser:	AdamW (fused)
β_1, β_2:	0.9, 0.95
ϵ:	10^{-8}
Base learning rate:	6×10^{-4}
Weight decay:	0.1 until step 94k 0.0 after step 94k
Scheduler:	Linear warmup + linear decay *(*)*
Total steps:	200k steps
Warm-up ratio:	1% of total steps (2k steps)
Final LR ratio:	1% of peak LR (6×10^{-6})
(*) Manual downward adjustments when observing long plateaus	
Micro batch size:	16 sequences
Gradient accumulation:	32 micro-steps until step 36k 64 micro-steps after step 36k
Precision:	`bfloat16`
`torch.compile`:	enabled

Data

Dataset:	OpenWebText (downloaded via Hugging Face `datasets`)
Tokeniser:	GPT-2 BPE tokeniser (vocabulary size 50,257)
Context length:	1,024 tokens
Validation split:	5,000 documents held out from training

Loss and Stats

Training:	loss, learning rate, and throughput logged every 50 steps
Validation:	loss computed every 1k steps over 2k randomly-sampled batches of size 4

Q.2 Results

Figure 3 shows the training cross-entropy loss (raw and exponential moving average with $\beta = 0.99$), the validation loss, and the validation perplexity over the course of training.

The next table reports the key metrics at the start, end, and best checkpoint of the run.

Step	Val. loss	Val. perplexity	Checkpoint
1,000	4.558	95.4	first evaluation
50,000	3.254	25.9	mid-training
99,500	**3.124**	**22.7**	**best checkpoint**
99,750	3.131	22.9	last evaluation

The average throughput was ~95,500 tokens/s.

Discussion. The training loss decreased from ~9.59 (step 50) to ~3.09 (step 99,975). The validation loss converged steadily throughout training, reaching its lowest value of 3.124 (perplexity 22.73) at step 99,500. The validation loss of 3.124 translates to the following CI.

$$\boxed{\text{true best loss} = 3.124 \pm 0.006\ , \qquad \text{with probability at least } 0.95\ .}$$

The CI is obtained as follows. With $k = 9$, for training steps after 97k, we obtain $\widehat{\mathrm{SE}} \approx 0.0025$. The pivot is t_t since we have $k - 2 = 7$ residual degrees of freedom, and hence the 95% CI is:

$$\hat{\mu} \ \pm \ t_{7,0.025} \cdot \widehat{\mathrm{SE}} \ = \ \hat{\mu} \pm 0.006\ .$$

The impact of doubling gradient accumulation from step 36k is clearly visible in the loss and perplexity curves. Also the impact of disabling weight decay in the last training phase (from step 94k) is visible in the loss and perplexity curves.

Although we used an LR schedule that would decay linearly in 200k training steps without our manual downwards adjustments, the results suggest that a linear or cosine decay in 100k steps would yield a similar outcome without any mid-training intervention—noting that they would have a similar trend to the learning rate curve in our run (see bottom plot in Figure 3).

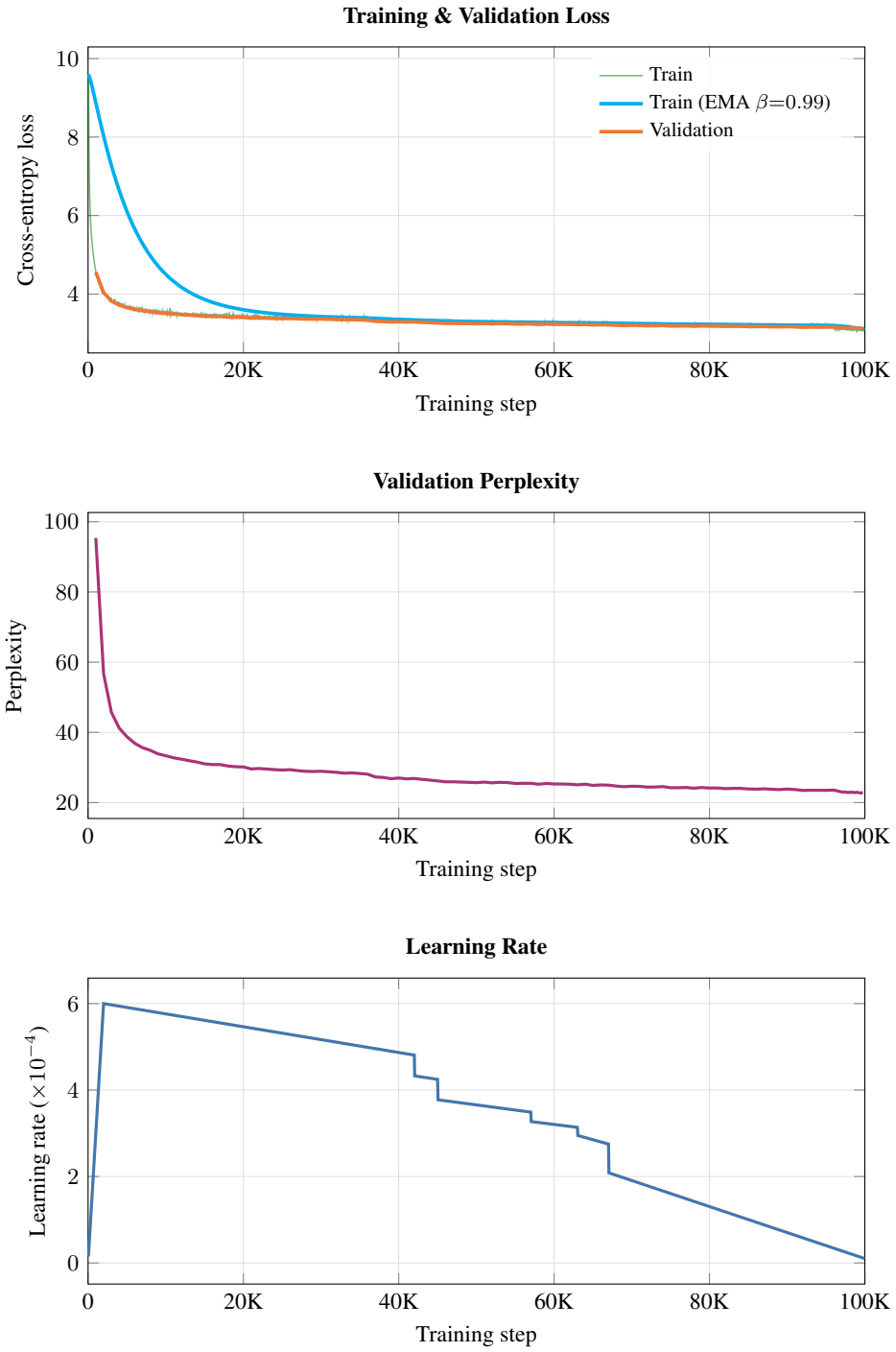


Figure 3: *Top:* training loss (EMA, $\beta = 0.99$) and validation loss. *Middle:* validation perplexity. *Bottom:* learning rate with linear warmup, linear decay, and manual downward adjustments.

Part V

Technical Analysis of Related Architectures

R Analysis of Linear RNNs

Table of Contents

R.1 Linear RNNs 272
R.2 General Linear Recurrent Units 273
R.3 Contractive Linear RNNs 284
R.4 Smoothness up to the Closure 286
R.5 Positive-Margin Expressivity of Smooth Contractive Linear RNNs 291
R.6 BIBS of Smooth Contractive Linear RNNs 293
R.7 BIBSD and nnVSD of Smooth Contractive Linear RNNs 293

R.1 Linear RNNs

Definition 255 (Linear Recurrent Units)**.** Given an input space $U \subseteq \mathbb{R}^{d_{\text{in}}}$, parameter space $\Theta_{\text{in}} = \Theta_A \times \Theta_B \subseteq \mathbb{R}^{d_A} \times \mathbb{R}^{d_B}$, and parametric functions

$$A : U \times \Theta_A \to \mathbb{R}^{d_{\text{state}} \times d_{\text{state}}}, \qquad B : U \times \Theta_B \to \mathbb{R}^{d_{\text{state}}},$$

a *parametric Linear Recurrent Unit* with input functions (A, B) is parametric dynamics $D = \langle X, U, f; \Theta \mid A, B \rangle$ having state space $X \subseteq \mathbb{R}^{d_{\text{state}}}$ and dynamics function given by the following equation:

$$f(x, u; \theta_A, \theta_B) \;=\; A(u; \theta_A) \cdot x + B(u; \theta_B)\,. \tag{168}$$

In the above Equation (168), the matrix $A(u; \theta_A) \in \mathbb{R}^{d_{\text{state}} \times d_{\text{state}}}$ is the *state-update matrix* on input u with parameter values θ_A and the vector $B(u; \theta_B) \in \mathbb{R}^{d_{\text{state}}}$ is the *additive term* on input u with parameter values θ_B.

A *Linear Recurrent Unit* is an instance $D[\theta]$ for $\theta \in \Theta$.

We denote by $\mathfrak{D}^{\text{p}}_{\text{Linear}}$ and $\mathfrak{D}_{\text{Linear}}$ the class of parametric Linear Recurrent Units and the class of Linear Recurrent Units, respectively. *End of Definition 255.*

Definition 256 (Linear Recurrent Layers)**.** *Linear Recurrent Layer dynamics* are recurrent layer $\mathfrak{D}_{\text{Linear}}$-dynamics.

Parametric Linear Recurrent Layer dynamics are recurrent layer $\mathfrak{D}^{\text{p}}_{\text{Linear}}$-dynamics.

Given a class $\mathfrak{G}$ of functions, A *Linear Recurrent Layer of type* $\mathfrak{G}$ is a recurrent layer of type $(\mathfrak{D}_{\text{Linear}}, \mathfrak{G})$.

Given a class $\mathfrak{G}^{\text{p}}$ of parametric functions, a *Parametric Linear Recurrent Layer of type* $\mathfrak{G}$ is a recurrent layer of type $(\mathfrak{D}^{\text{p}}_{\text{Linear}}, \mathfrak{G}^{\text{p}})$. *End of Definition 256.*

Definition 257 (Linear Recurrent Cascades)**.** Given a class $\mathfrak{G}$ of functions, *Linear Recurrent Cascade dynamics* are recurrent $\mathfrak{G}$-cascade $\mathfrak{D}_{\text{Linear}}$-dynamics, and a *Linear Recurrent Cascade of type* $\mathfrak{G}$ is a system having recurrent $\mathfrak{G}$-cascade $\mathfrak{D}_{\text{Linear}}$-dynamics and output function belonging to $\mathfrak{G}$.

Given a class $\mathfrak{G}^{\text{p}}$ of parametric functions, *Parametric Linear Recurrent Cascade dynamics* are recurrent $\mathfrak{G}^{\text{p}}$-cascade $\mathfrak{D}^{\text{p}}_{\text{Linear}}$-dynamics, and a *Parametric Linear Recurrent Cascade of type* $\mathfrak{G}$ is a system having recurrent $\mathfrak{G}^{\text{p}}$-cascade $\mathfrak{D}^{\text{p}}_{\text{Linear}}$-dynamics and output function belonging to $\mathfrak{G}^{\text{p}}$.
End of Definition 257.

Definition 258 (Linear Recurrent Neural Cascades)**.** *Linear Recurrent Neural Cascade dynamics*, abbreviated as *Linear RNC dynamics*, are $\mathfrak{D}_{\text{Linear}}$-RNC dynamics.

Parametric Linear Recurrent Neural Cascade dynamics, abbreviated as *Parametric Linear RNC dynamics*, are $\mathfrak{D}^{\text{p}}_{\text{Linear}}$-RNC dynamics.

A *Linear Recurrent Neural Cascade*, abbreviated as *Linear RNC*, is a $\mathfrak{D}_{\text{Linear}}$-RNC.

A *Parametric Linear Recurrent Neural Cascade*, abbreviated as *Parametric Linear RNC*, is a $\mathfrak{D}^{\text{p}}_{\text{Linear}}$-RNC. *End of Definition 258.*

Remark 46. We refer to *Linear RNCs* as *Linear RNNs*, and to *Parametric Linear RNCs* as *Parametric Linear RNNs*. The rationale is that, both in the research literature and in practice, the existing recurrent neural architectures are referred to as RNNs even though they are RNCs.

R.2 General Linear Recurrent Units

We show next that a general version of Linear Recurrent Units can be expressed as a Linear Recurrent Unit as per the definitions above, while preserving the norm of state updates.

Definition 259 (Linear operators)**.** We denote by $\mathbb{L}\big(\mathbb{R}^{m\times n}, \mathbb{R}^{m\times n}\big)$ the vector space of linear operators from $\mathbb{R}^{m\times n}$ to $\mathbb{R}^{m\times n}$. *End of Definition 259.*

Definition 260 (Frobenius norm of a linear operator)**.** Let $L \in \mathbb{L}\big(\mathbb{R}^{m\times n}, \mathbb{R}^{p\times q}\big)$.

For $i \in [1..m]$ and $j \in [1..n]$, let $E_{i,j} \in \mathbb{R}^{m\times n}$ denote the matrix having entry 1 at position (i, j) and entry 0 at every other position.

The *Frobenius norm* of L is

$$\|L\|_{\mathrm{F}} := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \|L(E_{i,j})\|_{\mathrm{F}}^2}.$$

End of Definition 260.

Definition 261 (General Linear Recurrent Units)**.** Given an input space $U \subseteq \mathbb{R}^{d_{\text{in}}}$ and functions

$$L : U \to \mathbb{L}\big(\mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}}\big), \qquad B : U \to \mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}},$$

a *General Linear Recurrent Unit* with input functions (L, B) is dynamics

$$D = \langle X, U, f \,|\, L, B\rangle, \qquad X \subseteq \mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}}$$

with function of the form

$$f(x, u) \;=\; L_u(x) + B(u) \qquad \text{with} \qquad L_u(x) := L(u)\,, \tag{169}$$

The linear operator L_u is called the *state-update linear operator* on input u, and the matrix $B(u)$ is called the *additive term* on input u. *End of Definition 261.*

Definition 262 (Parametric General Linear Recurrent Units)**.** Given Euclidean subspaces

$$U \subseteq \mathbb{R}^{d_{\text{in}}}, \quad \Theta_L \subseteq \mathbb{R}^{d_{\Theta_L}}, \quad \Theta_B \subseteq \mathbb{R}^{d_{\Theta_B}},$$

and parametric functions

$$L : \big(U \times \Theta_L\big) \to \mathbb{L}\big(\mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}}\big), \qquad B : \big(U \times \Theta_B\big) \to \mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}},$$

a *Parametric General Linear Recurrent Unit* with parametric input functions (L, B) is parametric dynamics

$$D = \langle X, U, f; \Theta_L, \Theta_B\rangle, \qquad X \subseteq \mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}},$$

with parametric dynamics function of the form

$$f(x, u; \theta_L, \theta_B) \;=\; L_u(x; \theta_L) + B(u; \theta_B) \qquad \text{with} \qquad L_u(x; \theta_L) := L(u; \theta_L)\,. \tag{170}$$

The linear operator $L_u(x; \theta_L)$ is called the *state-update linear operator* on input u with parameter values θ_L, and the matrix $B(u; \theta_B)$ is called the *additive term* on input u with parameter values θ_B. *End of Definition 262.*

Definition 263 (Left-multiplicative Linear Recurrent Units)**.** Given an input space $U \subseteq \mathbb{R}^{d_{\text{in}}}$, and functions

$$A : U \to \mathbb{R}^{d_{\text{state}}\times d_{\text{state}}}, \qquad B : U \to \mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}},$$

a *Left-Multiplicative Linear Recurrent Unit* with input functions (A, B) is dynamics

$$D = \langle X, U, f \,|\, A, B\rangle, \qquad X \subseteq \mathbb{R}^{d_{\text{state}}\times n_{\text{heads}}},$$

with dynamics function of the form

$$f(H, u) \;:=\; A(u) \cdot H + B(u).$$

The matrix $A(u)$ is called the *(left-multiplicative) state-update matrix* on input u, and the matrix $B(u)$ is called the *additive term* on input u. *End of Definition 263.*

Definition 264 (Parametric left-multiplicative linear recurrent units)**.** Given Euclidean subspaces

$$U \subseteq \mathbb{R}^{d_{\text{in}}}, \quad \Theta_A \subseteq \mathbb{R}^{d_{\Theta_A}}, \quad \Theta_B \subseteq \mathbb{R}^{d_{\Theta_B}},$$

and parametric functions

$$A : \big(U \times \Theta_A\big) \to \mathbb{R}^{d_{\text{state}} \times d_{\text{state}}}, \qquad B : \big(U \times \Theta_B\big) \to \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}},$$

a *parametric left-multiplicative linear recurrent unit* with input functions (A, B) is dynamics

$$D = \langle X, U, f; A, B \,|\, \Theta_A, \Theta_B \rangle, \qquad X \subseteq \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}},$$

with parametric dynamics function of the form

$$f(H, u; \theta_A, \theta_B) \;:=\; A(u; \theta_A) \cdot H + B(u; \theta_B).$$

The matrix $A(u; \theta_A)$ is called the *(left-multiplicative) state-update matrix* on input u for parameter values θ_A, and the matrix $B(u; \theta_B)$ is called the *additive term* on input u for parameter values θ_B.
End of Definition 264.

R.2.1 Facts about General Linear Recurrent Units

We next provide facts about the above definitions.

Proposition 61 (Correctness of instantiation of Parametric General Linear Recurrent Units)**.** *Let D be a Parametric General Linear Recurrent Unit, and let Θ be its joint parameter space. For every $\theta \in \Theta$, we have that $D[\theta]$ is a General Linear Recurrent Unit.* End of Proposition 61.

Proof. Let

$$D = \langle X, U, f; \Theta \rangle$$

be a Parametric General Linear Recurrent Unit, with parametric input functions

$$L : U \times \Theta_L \to \mathbb{L}\big(\mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}\big) \qquad \text{and} \qquad B : U \times \Theta_B \to \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}},$$

where

$$\Theta = \Theta_L \times \Theta_B.$$

Let

$$\theta = (\theta_L, \theta_B) \in \Theta.$$

The instance $D[\theta]$ has instantiated input functions

$$\begin{aligned} L[\theta_L](u) &:= L(u, \theta_L), \\ B[\theta_B](u) &:= B(u, \theta_B), \end{aligned}$$

and instantiated dynamics function

$$\begin{aligned} f[\theta](x, u) &:= f(x, u; \theta) \\ &= L(u, \theta_L)(x) + B(u, \theta_B) \\ &= L[\theta_L](u)(x) + B[\theta_B](u). \end{aligned}$$

Furthermore,

$$L[\theta_L] : U \to \mathbb{L}\big(\mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}\big)$$

and

$$B[\theta_B] : U \to \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}.$$

Hence $D[\theta]$ satisfies the definition of a General Linear Recurrent Unit. □

Proposition 62 (Correctness of instantiation of parametric left-multiplicative linear recurrent units)**.** *Let D be a parametric Left-Multiplicative Linear Recurrent Unit, and let Θ be its joint parameter space. For every $\theta \in \Theta$, we have that $D[\theta]$ is a Left-Multiplicative Linear Recurrent Unit.*
End of Proposition 62.

Proof. Let

$$D = \langle X, U, f; A, B \,|\, \Theta_A, \Theta_B \rangle$$

be a parametric Left-Multiplicative Linear Recurrent Unit, where

$$\Theta = \Theta_A \times \Theta_B$$

is its joint parameter space, and

$$A : U \times \Theta_A \to \mathbb{R}^{d_{\text{state}} \times d_{\text{state}}},$$
$$B : U \times \Theta_B \to \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}$$

are its parametric input functions.

Let

$$\theta = (\theta_A, \theta_B) \in \Theta.$$

The instance $D[\theta]$ has instantiated input functions

$$A[\theta_A](u) := A(u, \theta_A),$$
$$B[\theta_B](u) := B(u, \theta_B),$$

and instantiated dynamics function

$$\begin{aligned} f[\theta](H, u) &:= f(H, u; \theta_A, \theta_B) \\ &= A(u, \theta_A) \cdot H + B(u, \theta_B) \\ &= A[\theta_A](u) \cdot H + B[\theta_B](u). \end{aligned}$$

Furthermore,

$$A[\theta_A] : U \to \mathbb{R}^{d_{\text{state}} \times d_{\text{state}}}$$

and

$$B[\theta_B] : U \to \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}.$$

Hence $D[\theta]$ satisfies the definition of a Left-Multiplicative Linear Recurrent Unit. □

Fact 58 (Left-multiplicative Linear Recurrent Units are General Linear Recurrent Units)**.** *We have that (parametric) left-multiplicative Linear Recurrent Units are (parametric, resp.) General Linear Recurrent Units, as more explicitly stated below.*

Fact 58.1: Non-parametric case. *Every Left-Multiplicative Linear Recurrent Unit is a General Linear Recurrent Unit. Specifically, every Left-Multiplicative Linear Recurrent Unit*

$$D = \langle X, U, f \,|\, A, B \rangle, \qquad X \subseteq \Omega_X = \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}},$$

can be equivalently expressed as the General Linear Recurrent Unit

$$\tilde{D} := \langle X, U, f \,|\, L^A, B \rangle$$

where the function $L^A : U \to \mathbb{L}(\Omega_X)$ is given by

$$L^A(u) := L^A_u \quad \textit{with} \quad L^A_u(H) := A(u) \cdot H \qquad \forall\, u \in U,\ H \in \Omega_X. \tag{171}$$

Fact 58.2: Parametric case. *Every Parametric Left-Multiplicative Linear Recurrent Unit is a Parametric General Linear Recurrent Unit. Specifically, every Parametric Left-Multiplicative Linear Recurrent Unit*

$$D = \langle X, U, f; A, B \,|\, \Theta_A, \Theta_B \rangle, \qquad X \subseteq \Omega_X = \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}},$$

can be equivalently expressed as the Parametric General Linear Recurrent Unit

$$\tilde{D} := \langle X, U, f; L^A, B \,|\, \Theta_A, \Theta_B \rangle$$

where the parametric function $L^A : \big(U \times \Theta_A\big) \to \mathbb{L}(\Omega_X)$ is given by

$$L^A(u; \theta_A) := L^A_u[\theta_A] \ \textit{ with } \ L^A_u(H; \theta_A) := A(u; \theta_A) \cdot H \qquad \forall\, \theta_A \in \Theta_A, u \in U, H \in \Omega_X. \tag{172}$$

End of Fact 58.

Proof of Fact 58.1: Non-parametric case. Let

$$D = \langle X, U, f \mid A, B \rangle$$

be a Left-Multiplicative Linear Recurrent Unit, with

$$X \subseteq \Omega_X = \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}.$$

Let

$$L^A : U \to \mathbb{L}(\Omega_X)$$

be defined as in Equation (171).

We first verify that L^A is well defined. Fix $u \in U$. For every $H_1, H_2 \in \Omega_X$ and $\alpha, \beta \in \mathbb{R}$,

$$\begin{aligned} L^A_u\big(\alpha H_1 + \beta H_2\big) &= A(u) \cdot \big(\alpha H_1 + \beta H_2\big) \\ &= \alpha A(u) \cdot H_1 + \beta A(u) \cdot H_2 \\ &= \alpha L^A_u(H_1) + \beta L^A_u(H_2). \end{aligned}$$

Hence L^A_u is a linear operator on Ω_X. Since $u \in U$ was arbitrary,

$$L^A : U \to \mathbb{L}(\Omega_X)$$

is well defined.

It remains to verify that $\tilde{D}$ has the same dynamics as D. Since D is a Left-Multiplicative Linear Recurrent Unit, for every $H \in X$ and $u \in U$,

$$\begin{aligned} f(H, u) &= A(u) \cdot H + B(u) \\ &= L^A_u(H) + B(u) \\ &= L^A(u)(H) + B(u). \end{aligned}$$

Therefore,

$$\tilde{D} = \langle X, U, f \mid L^A, B \rangle$$

satisfies the definition of a General Linear Recurrent Unit, and has the same state space, input space, and dynamics function as D. □

Proof of Fact 58.2: Parametric case. Let

$$D = \langle X, U, f; A, B \mid \Theta_A, \Theta_B \rangle$$

be a Parametric Left-Multiplicative Linear Recurrent Unit, with

$$X \subseteq \Omega_X = \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}.$$

Let

$$L^A : U \times \Theta_A \to \mathbb{L}(\Omega_X)$$

be defined as in Equation (172).

We first verify that L^A is well defined. Fix $u \in U$ and $\theta_A \in \Theta_A$. For every $H_1, H_2 \in \Omega_X$ and $\alpha, \beta \in \mathbb{R}$,

$$\begin{aligned} L^A_u(\alpha H_1 + \beta H_2; \theta_A) &= A(u; \theta_A) \cdot \big(\alpha H_1 + \beta H_2\big) \\ &= \alpha A(u; \theta_A) \cdot H_1 + \beta A(u; \theta_A) \cdot H_2 \\ &= \alpha L^A_u(H_1; \theta_A) + \beta L^A_u(H_2; \theta_A). \end{aligned}$$

Hence $L^A_u[\theta_A]$ is a linear operator on Ω_X. Since $u \in U$ and $\theta_A \in \Theta_A$ were arbitrary,

$$L^A : U \times \Theta_A \to \mathbb{L}(\Omega_X)$$

is well defined.

It remains to verify that $\tilde{D}$ has the same parametric dynamics as D. Since D is a Parametric Left-Multiplicative Linear Recurrent Unit, for every $H \in X$, $u \in U$, $\theta_A \in \Theta_A$, and $\theta_B \in \Theta_B$,

$$\begin{aligned} f(H, u; \theta_A, \theta_B) &= A(u; \theta_A) \cdot H + B(u; \theta_B) \\ &= L^A_u(H; \theta_A) + B(u; \theta_B) \\ &= L^A(u; \theta_A)(H) + B(u; \theta_B). \end{aligned}$$

Therefore,

$$\tilde{D} = \langle X, U, f; L^A, B \mid \Theta_A, \Theta_B \rangle$$

satisfies the definition of a Parametric General Linear Recurrent Unit, and has the same state space, input space, parameter spaces, and parametric dynamics function as D. □

R.2.2 Reduction to Linear Recurrent Units

Definition 265. Let $m, n, p, q \in \mathbb{N}^{>0}$.

▸ **SubDefinition 265.1**. The *flattening operator* on matrices is the linear bijection $\mathrm{flatten} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ that maps every matrix to the vector obtained by stacking its columns; namely, it is the map

$$\begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{pmatrix} \quad \mapsto \quad \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{m,1} \\ \vdots \\ x_{1,n} \\ \vdots \\ x_{m,n} \end{pmatrix}.$$

End of SD.265.1 ◂

▸ **SubDefinition 265.2**. For every subset $X \subseteq \mathbb{R}^{m\times n}$, we define its flattening element-wise as

$$\mathrm{flatten}(X) := \big\{\mathrm{flatten}(x) \mid x \in X\big\} \subseteq \mathbb{R}^{mn}.$$

End of SD.265.2 ◂

▸ **SubDefinition 265.3**. The flattening operator on linear operators is the map

$$\mathrm{flatten} : \mathbb{L}\big(\mathbb{R}^{m\times n}, \mathbb{R}^{p\times q}\big) \to \mathbb{R}^{(pq)\times(mn)}$$

such that, for every linear operator

$$L \in \mathbb{L}\big(\mathbb{R}^{m\times n}, \mathbb{R}^{p\times q}\big),$$

we have that $\mathrm{flatten}(L)$ is the unique matrix satisfying

$$\mathrm{flatten}\big(L(X)\big) \;=\; \mathrm{flatten}(L)\cdot \mathrm{flatten}(X) \qquad\qquad \forall X \in \mathbb{R}^{m\times n}.$$

End of SD.265.3 ◂

End of Definition 265.

Definition 266 (State flattening induced by recurrent cascade presentations)**.** Let us consider a parametric recurrent cascade presentation

$$\mathsf{C} \;=\; g_1 \blacktriangleright \mathsf{L}_1 \blacktriangleleft_{g_2} \mathsf{L}_2 \blacktriangleleft_{g_3} \cdots \blacktriangleleft_{g_\ell} \mathsf{L}_\ell \qquad \text{where} \qquad \mathsf{L}_i \;=\; D_{i,1} \parallel \cdots \parallel D_{i,m_i} \quad \forall i \in [1..\ell]\,.$$

▸ **SubDefinition 266.1**. Let $i \in [1..\ell]$ be a layer index, let $j \in [1..m_i]$ be an index of a unit within the i-th layer, and let $X_{i,j}$ the state space of unit $D_{i,j}$.

Assume that the ambient space $\Omega_{X_{i,j}}$ of the state space $X_{i,j}$ is either $\mathbb{R}^{d_{i,j}}$ or $\mathbb{R}^{r_{i,j}\times c_{i,j}}$.

The *flattened ambient state space* of unit $D_{i,j}$ is

$$\tilde{\Omega}_{X_{i,j}} \;:=\; \begin{cases} \mathbb{R}^{d_{i,j}} & \text{if } \Omega_{X_{i,j}} = \mathbb{R}^{d_{i,j}}, \\ \mathbb{R}^{r_{i,j}c_{i,j}} & \text{if } \Omega_{X_{i,j}} = \mathbb{R}^{r_{i,j}\times c_{i,j}}\,. \end{cases}$$

The *state flattening* for unit $D_{i,j}$ is the linear bijection

$$\phi_{i,j} : \Omega_{X_{i,j}} \to \tilde{\Omega}_{X_{i,j}}, \qquad\qquad \phi_{i,j} \;:=\; \begin{cases} \mathrm{id}_{\mathbb{R}^{d_{i,j}}} & \text{if } \Omega_{X_{i,j}} = \mathbb{R}^{d_{i,j}}, \\ \mathrm{flatten} & \text{if } \Omega_{X_{i,j}} = \mathbb{R}^{r_{i,j}\times c_{i,j}}\,. \end{cases}$$

The *flattened state space* of unit $D_{i,j}$ is

$$\tilde{X}_{i,j} \;:=\; \phi_{i,j}(X_{i,j})\,.$$

End of SD.266.1 ◂

▸ **SubDefinition 266.2**. Let $i \in [1..\ell]$ be a layer index.

The *state flattening at layer* i induced by C is

$$\phi_i \;:=\; \big(\phi_{i,1}, \ldots, \phi_{i,m_i}\big)\,.$$

For $i \geq 2$, the *prefix state flattening up to layer* $i-1$ induced by C is

$$\phi_{<i} \;:=\; \big(\phi_1, \ldots, \phi_{i-1}\big)\,.$$

End of SD.266.2 ◂

▸ **SubDefinition 266.3**. The *state flattening induced by* C is

$$\phi_{\mathsf{C}} \;:=\; \big(\phi_1,\, \ldots,\, \phi_\ell\big),$$

and the *flattened state space induced by* C is

$$\tilde{X} \;:=\; \phi_{\mathsf{C}}(X)$$

for X the state space of $[\![\mathsf{C}]\!]$. *End of SD.266.3* ◂

End of Definition 266.

Fact 59 (Invertibility of presentation-induced state flattening)**.** *In the context of Definition 266, the maps* $(\phi_i)_{i=1}^{\ell}$, $(\phi_{<i})_{i=2}^{\ell}$, *and* ϕ_{C} *are linear bijections. In particular,*

$$\phi_{\mathsf{C}}^{-1} \;=\; \big(\phi_1^{-1},\, \ldots,\, \phi_\ell^{-1}\big) \qquad \textit{where} \qquad \phi_i^{-1} \;=\; \big(\phi_{i,1}^{-1},\, \ldots,\, \phi_{i,m_i}^{-1}\big) \quad \forall i \in [1..\ell].$$

End of Fact 59.

Proof. Each $\phi_{i,j}$ is either an identity map or the matrix flattening map of Definition 265, and hence is a linear bijection. Finite Cartesian products of linear bijections are linear bijections, with inverse given component-wise. □

Definition 267 (Canonical Linear Recurrent Unit representation)**.** Let us consider a Parametric General Linear Recurrent Unit

$$D = \langle X, U, f; \Theta_L, \Theta_B \rangle,$$

let its state space be $X \subseteq \Omega_X = \mathbb{R}^{d_{\text{state}} \times n_{\text{heads}}}$, and let its parametric input functions be

$$L : U \times \Theta_L \to \mathbb{L}\big(\Omega_X, \Omega_X\big) \qquad \text{and} \qquad B : U \times \Theta_B \to \Omega_X.$$

Let us define

$$d \;:=\; d_{\text{state}} \cdot n_{\text{heads}}\,, \qquad \phi \;:=\; \text{flatten} : \Omega_X \to \mathbb{R}^d\,, \qquad \tilde{X} \;:=\; \phi(X)\,.$$

The *canonical Linear Recurrent Unit representation of* D is the Parametric Linear Recurrent Unit

$$\tilde{D} \;:=\; \langle \tilde{X}, U, \tilde{f}; \Theta_L \times \Theta_B \,|\, \tilde{A}, \tilde{B} \rangle$$

whose parametric input functions are

$$\tilde{A}(u; \theta_L) \;:=\; \text{flatten}\,(L(u; \theta_L)) \qquad \text{and} \qquad \tilde{B}(u; \theta_B) \;:=\; \text{flatten}\,(B(u; \theta_B))\,.$$

End of Definition 267.

Remark 47. In the context of Definition 267, the dynamics function $\tilde{f}$ is given by

$$\tilde{f}\big(\tilde{x}, u; \theta_L, \theta_B\big) \;=\; \tilde{A}(u; \theta_L) \cdot \tilde{x} + \tilde{B}(u; \theta_B).$$

End of Remark 47.

Proposition 63 (Correctness of the canonical Linear Recurrent Unit representation)**.** *In the context of Definition 267, we have*

$$\phi\left(f\big(x, u; \theta_L, \theta_B\big)\right) \;=\; \tilde{f}\big(\phi(x), u; \theta_L, \theta_B\big) \qquad \forall x \in X,\ \forall u \in U,\ \forall \theta_L \in \Theta_L,\ \forall \theta_B \in \Theta_B.$$

Hence D *and* $\tilde{D}$ *are isomorphic under the state isomorphism* ϕ. End of Proposition 63.

Proof. By the definition of a Parametric General Linear Recurrent Unit,

$$\begin{aligned}\phi\left(f\big(x,u;\,\theta_L,\theta_B\big)\right) &= \operatorname{flatten}\left(L(u;\theta_L)(x)+B(u;\theta_B)\right)\\ &= \operatorname{flatten}\left(L(u;\theta_L)(x)\right)+\operatorname{flatten}\left(B(u;\theta_B)\right).\end{aligned}$$

By Definition 265,

$$\operatorname{flatten}\left(L(u;\theta_L)(x)\right) = \operatorname{flatten}\left(L(u;\theta_L)\right)\cdot\operatorname{flatten}(x).$$

Therefore,

$$\begin{aligned}\phi\left(f\big(x,u;\,\theta_L,\theta_B\big)\right) &= \tilde{A}(u;\theta_L)\cdot\phi(x)+\tilde{B}(u;\theta_B)\\ &= \tilde{f}\big(\phi(x),u;\,\theta_L,\theta_B\big),\end{aligned}$$

proving the fact. □

Definition 268 (Linear RNN representations induced by RNC presentations)**.** Let us consider a parametric RNC

$$S = \langle X, U_S, f_S, x^{\text{init}}, Y, h;\,\Theta_f,\Theta_h\rangle,$$

and a parametric analytic output extension presentation of S given by

$$\mathsf{S} \;:=\; \langle \mathsf{C}, x^{\text{init}}\rangle \blacktriangleright h.$$

Suppose that the above parametric recurrent cascade presentation C is the form

$$\mathsf{C} \;=\; g_1 \blacktriangleright \mathsf{L}_1 \blacktriangleright\!\!\!\prec_{g_2} \mathsf{L}_2 \blacktriangleright\!\!\!\prec_{g_3} \cdots \blacktriangleright\!\!\!\prec_{g_\ell} \mathsf{L}_\ell$$

where, in turn, each parametric recurrent layer presentation L_i is of the form

$$\mathsf{L}_i \;=\; D_{i,1} \parallel \cdots \parallel D_{i,m_i}$$

with units $D_{i,1},\ldots,D_{i,m_i}$ being Parametric General Linear Recurrent Units.

Let $(\phi_i)i = 1^\ell$, $(\phi_{<i})_{i=2}^{\ell}$, and ϕ_{C} be the state flattenings induced by C according to Definition 266.

▸ **SubDefinition 268.1**. Let $i \in [1..\ell]$ be a layer index. For every $j \in [1..m_i]$, let us write $\tilde{D}_{i,j}$ for the canonical Linear Recurrent Unit representation of $D_{i,j}$ according to Definition 267.

We define *flattened layer presentation for layer* i as

$$\tilde{\mathsf{L}}_i \;:=\; \tilde{D}_{i,1} \parallel \cdots \parallel \tilde{D}_{i,m_i}\,.$$

End of SD.268.1 ◂

▸ **SubDefinition 268.2**. Let $i \in [2..\ell]$ be a layer index—note that $i \geq 2$.

We define the *flattened i-th connecting function* as

$$\tilde{g}_i\big(\tilde{x}_{\text{pre}},u,\tilde{x}_{\text{cur}};\,\theta_{g_i}\big) \;:=\; g_i\Big(\phi_{<i}^{-1}(\tilde{x}_{\text{pre}}),\;u,\;\phi_{<i}^{-1}(\tilde{x}_{\text{cur}});\;\theta_{g_i}\Big). \tag{173}$$

End of SD.268.2 ◂

▸ **SubDefinition 268.3**. Given the above intermediate definitions, we define the following.

The *flattened recurrent cascade presentation* associated with C is

$$\tilde{\mathsf{C}} \;:=\; g_1 \blacktriangleright \tilde{\mathsf{L}}_1 \blacktriangleright\!\!\!\prec_{\tilde{g}_2} \tilde{\mathsf{L}}_2 \blacktriangleright\!\!\!\prec_{\tilde{g}_3} \cdots \blacktriangleright\!\!\!\prec_{\tilde{g}_\ell} \tilde{\mathsf{L}}_\ell.$$

The *flattened initial state* associated with C is

$$\tilde{x}^{\text{init}} \;:=\; \phi_{\mathsf{C}}\big(x^{\text{init}}\big)\,.$$

The *flattened output function* associated with C is

$$\tilde{h}\Big(\tilde{x}_{\text{pre}},\;u,\;\tilde{x}_{\text{cur}};\;\theta_h\Big) \;:=\; h\Big(\phi_{\mathsf{C}}^{-1}\big(\tilde{x}_{\text{pre}}\big),\;u,\;\phi_{\mathsf{C}}^{-1}\big(\tilde{x}_{\text{cur}}\big);\;\theta_h\Big)\,. \tag{174}$$

The *Linear RNN presentation* associated with S is the output extension presentation

$$\tilde{\mathsf{S}} \;:=\; \langle \tilde{\mathsf{C}}, \tilde{x}^{\mathrm{init}} \rangle \blacktriangleright \tilde{h}.$$

The *Linear RNN representation of S* associated with S is the parametric system

$$\tilde{S} \;:=\; [\![\tilde{\mathsf{S}}]\!].$$

End of SD.268.3 ◂

End of Definition 268.

Proposition 64 (Correctness of presentation-induced Linear RNN representations)**.** *Let*

$$S = \langle X, U, f, x^{\mathrm{init}}, Y, h; \Theta_f, \Theta_h \rangle$$

be a parametric RNC, and let

$$\mathsf{S} \;:=\; \langle \mathsf{C}, x^{\mathrm{init}} \rangle \blacktriangleright h$$

be a parametric analytic output extension presentation of S whose recurrent cascade presentation is of the form

$$\mathsf{C} \;=\; g_1 \blacktriangleright \mathsf{L}_1 \blacktriangleleft_{g_2} \mathsf{L}_2 \blacktriangleleft_{g_3} \cdots \blacktriangleleft_{g_\ell} \mathsf{L}_\ell \qquad \textit{where} \qquad \mathsf{L}_i \;=\; D_{i,1} \parallel \cdots \parallel D_{i,m_i} \qquad \forall i \in [1..\ell],$$

and each $D_{i,j}$ is a Parametric General Linear Recurrent Unit.

Let $\tilde{\mathsf{C}}$ be the flattened recurrent cascade presentation, let $\tilde{x}^{\mathrm{init}}$ be the flattened initial state, let $\tilde{h}$ be the flattened output function, let $\tilde{\mathsf{S}}$, be the associated Linear RNN presentation, and let $\tilde{S}$ be the associated Linear RNN representation—all induced by S according to Definition 268.

Let ϕ_{C} be the state flattening induced by C—according to Definition 266.

Let us write

$$[\![\mathsf{C}]\!] = \langle X, U, f; \Theta_f \rangle, \qquad\qquad [\![\tilde{\mathsf{C}}]\!] = \langle \tilde{X}, U, \tilde{f}; \Theta_f \rangle.$$

Then:

- $\tilde{X} = \phi_{\mathsf{C}}(X)$*;*
- *for every $x \in X$, every $u \in U$, and every $\theta_f \in \Theta_f$,*

$$\phi_{\mathsf{C}}\left(f(x, u; \theta_f)\right) = \tilde{f}\big(\phi_{\mathsf{C}}(x), u; \theta_f\big). \tag{175}$$

Furthermore:

- $\tilde{x}^{\mathrm{init}} = \phi_{\mathsf{C}}(x^{\mathrm{init}})$*;*
- *for every $x_{\mathrm{pre}}, x_{\mathrm{cur}} \in X$, every $u \in U$, and every $\theta_h \in \Theta_h$,*

$$\tilde{h}\big(\phi_{\mathsf{C}}(x_{\mathrm{pre}}), u, \phi_{\mathsf{C}}(x_{\mathrm{cur}});\, \theta_h\big) = h\big(x_{\mathrm{pre}}, u, x_{\mathrm{cur}}; \theta_h\big). \tag{176}$$

Consequently, S and $\tilde{S}$ have the same input-output behaviour under the same parameter values; namely,

$$S[\theta](u) \;=\; \tilde{S}[\theta](u) \qquad\qquad \forall \theta \in \Theta_f \times \Theta_h, \quad \forall u \in U^+.$$

End of Proposition 64.

Proof. We first prove the claims concerning the cascade dynamics.

For every $i \in [1..\ell]$, let X_i and $\tilde{X}_i$ denote the state spaces of the dynamics presented by L_i and $\tilde{\mathsf{L}}_i$, respectively, and let f_i and $\tilde{f}_i$ denote their respective parametric dynamics functions.

By Proposition 63, for every $j \in [1..m_i]$, the unit state flattening $\phi_{i,j}$ maps the state space of $D_{i,j}$ onto the state space of $\tilde{D}_{i,j}$ and conjugates their parametric dynamics functions. Since parallel composition operates component-wise, it follows that

$$\tilde{X}_i = \phi_i(X_i) \tag{177}$$

and that

$$\phi_i\left(f_i(x_i, v; \theta_i)\right) = \tilde{f}_i\big(\phi_i(x_i), v; \theta_i\big) \tag{178}$$

for every admissible x_i, v, and corresponding layer parameter value θ_i.

We now prove the corresponding claims for the recurrent cascade by induction over its layers.

For $i \in [1..\ell]$, let $\mathsf{C}_{\le i}$ and $\tilde{\mathsf{C}}_{\le i}$ denote, respectively, the prefixes through layer i of C and $\tilde{\mathsf{C}}$. Furthermore, define

$$\phi_{\le i} \;:=\; \big(\phi_1, \ldots, \phi_i\big).$$

Thus, for $i \ge 2$,

$$\phi_{<i} = \phi_{\le i-1},$$

and

$$\phi_{\le \ell} = \phi_{\mathsf{C}}.$$

We show inductively that, for every $i \in [1..\ell]$, the state space $\tilde{X}_{\le i}$ presented by $\tilde{\mathsf{C}}_{\le i}$ satisfies

$$\tilde{X}_{\le i} = \phi_{\le i}(X_{\le i}), \tag{179}$$

where $X_{\le i}$ is the state space presented by $\mathsf{C}_{\le i}$, and that their parametric dynamics functions $f_{\le i}$ and $\tilde{f}_{\le i}$ satisfy

$$\phi_{\le i}\left(f_{\le i}(x, u; \theta_{\le i})\right) = \tilde{f}_{\le i}\big(\phi_{\le i}(x), u; \theta_{\le i}\big) \tag{180}$$

for every admissible state x, input u, and corresponding parameter value $\theta_{\le i}$.

For $i = 1$, the two prefix presentations are

$$g_1 \blacktriangleright \mathsf{L}_1 \qquad \text{and} \qquad g_1 \blacktriangleright \tilde{\mathsf{L}}_1.$$

They use the same input-precomposition function g_1. Therefore Equations (179) and (180) for $i = 1$ follow directly from Equations (177) and (178).

Let now $i \in [2..\ell]$, and suppose that Equations (179) and (180) hold for $i - 1$.

Consider an arbitrary state of the original prefix through layer i, written as

$$x = (x_{<i}, x_i),$$

where $x_{<i} \in X_{\le i-1}$ and $x_i \in X_i$. Its corresponding flattened state is

$$\tilde{x} = (\phi_{<i}(x_{<i}), \phi_i(x_i)) = \phi_{\le i}(x).$$

Fix an input u, and let

$$x'_{<i} \;:=\; f_{\le i-1}(x_{<i}, u; \theta_{<i})$$

be the updated state of the original prefix through layer $i - 1$. By the induction hypothesis, the corresponding updated flattened prefix state is

$$\tilde{x}'_{<i} = \phi_{<i}(x'_{<i}).$$

Let

$$v_i \;:=\; g_i\big(x_{<i}, u, x'_{<i}; \theta_{g_i}\big)$$

be the input presented to layer i in the original cascade. By Equation (173) and the preceding state correspondences, the input presented to the flattened layer i is

$$\begin{aligned}
\tilde{g}_i\big(\phi_{<i}(x_{<i}), u, \phi_{<i}(x'_{<i}); \theta_{g_i}\big) &= g_i\Big(\phi_{<i}^{-1}\big(\phi_{<i}(x_{<i})\big), u, \phi_{<i}^{-1}\big(\phi_{<i}(x'_{<i})\big);\; \theta_{g_i}\Big) \\
&= g_i\big(x_{<i}, u, x'_{<i}; \theta_{g_i}\big) \\
&= v_i.
\end{aligned}$$

Hence the original and flattened versions of layer i receive the same input. Applying Equation (178), their updated states satisfy

$$\phi_i\left(f_i(x_i, v_i; \theta_i)\right) = \tilde{f}_i\big(\phi_i(x_i), v_i; \theta_i\big).$$

Consequently, the complete updated states of the two prefixes correspond under

$$\phi_{\le i} = \big(\phi_{<i}, \phi_i\big),$$

proving Equation (180) for i. Furthermore, by the induction hypothesis and Equation (177),

$$\begin{aligned}
\tilde{X}_{\le i} &= \tilde{X}_{\le i-1} \times \tilde{X}_i \\
&= \phi_{<i}(X_{\le i-1}) \times \phi_i(X_i) \\
&= \phi_{\le i}\big(X_{\le i-1} \times X_i\big) \\
&= \phi_{\le i}(X_{\le i}),
\end{aligned}$$

proving Equation (179) for i.

The induction is therefore complete. Taking $i = \ell$, we obtain

$$\tilde{X} = \phi_{\mathsf{C}}(X),$$

together with

$$\phi_{\mathsf{C}}\left(f(x, u; \theta_f)\right) = \tilde{f}\big(\phi_{\mathsf{C}}(x), u; \theta_f\big),$$

for every $x \in X$, $u \in U$, and $\theta_f \in \Theta_f$. This proves Equation (175) and the first state-space claim.

We next consider the output extension. By Definition 268,

$$\tilde{x}^{\mathrm{init}} = \phi_{\mathsf{C}}\big(x^{\mathrm{init}}\big),$$

proving the claimed relation between the initial states. Moreover, by Equation (174), for every $x_{\mathrm{pre}}, x_{\mathrm{cur}} \in X$, every $u \in U$, and every $\theta_h \in \Theta_h$,

$$\begin{aligned}
\tilde{h}\big(\phi_{\mathsf{C}}(x_{\mathrm{pre}}), u, \phi_{\mathsf{C}}(x_{\mathrm{cur}});\, \theta_h\big) &= h\Big(\phi_{\mathsf{C}}^{-1}\big(\phi_{\mathsf{C}}(x_{\mathrm{pre}})\big), u, \phi_{\mathsf{C}}^{-1}\big(\phi_{\mathsf{C}}(x_{\mathrm{cur}})\big);\, \theta_h\Big) \\
&= h\big(x_{\mathrm{pre}}, u, x_{\mathrm{cur}};\, \theta_h\big).
\end{aligned}$$

This proves Equation (176).

It remains to prove equality of the represented input–output behaviours. Fix arbitrary

$$\theta = (\theta_f, \theta_h) \in \Theta_f \times \Theta_h$$

and

$$u = (u_t)_{t \in [T]} \in U^+.$$

Let

$$(x_t)_{t \in [0..T]} \qquad \text{and} \qquad (\tilde{x}_t)_{t \in [0..T]}$$

be the state sequences of $S[\theta]$ and $\tilde{S}[\theta]$, respectively.

We claim that

$$\tilde{x}_t = \phi_{\mathsf{C}}(x_t) \qquad \forall t \in [0..T]. \tag{181}$$

For $t = 0$, this follows from

$$\tilde{x}_0 = \tilde{x}^{\mathrm{init}} = \phi_{\mathsf{C}}\big(x^{\mathrm{init}}\big) = \phi_{\mathsf{C}}(x_0).$$

Suppose now that Equation (181) holds at time $t-1$. Then, by Equation (175),

$$\begin{aligned}
\tilde{x}_t &= \tilde{f}\big(\tilde{x}_{t-1}, u_t; \theta_f\big) \\
&= \tilde{f}\big(\phi_{\mathsf{C}}(x_{t-1}), u_t; \theta_f\big) \\
&= \phi_{\mathsf{C}}\left(f(x_{t-1}, u_t; \theta_f)\right) \\
&= \phi_{\mathsf{C}}(x_t).
\end{aligned}$$

Hence Equation (181) follows by induction.

Let $(y_t)_{t \in [T]}$ and $(\tilde{y}_t)_{t \in [T]}$ be the respective output sequences. By Equation (176) and Equation (181), for every $t \in [T]$,

$$\begin{aligned}
\tilde{y}_t &= \tilde{h}\big(\tilde{x}_{t-1}, u_t, \tilde{x}_t;\, \theta_h\big) \\
&= \tilde{h}\big(\phi_{\mathsf{C}}(x_{t-1}), u_t, \phi_{\mathsf{C}}(x_t);\, \theta_h\big) \\
&= h\big(x_{t-1}, u_t, x_t;\, \theta_h\big) \\
&= y_t.
\end{aligned}$$

Therefore,

$$S\check{[}\theta](u) = \tilde{}\check{[}\theta](u).$$

Since $\theta \in \Theta_f \times \Theta_h$ and $u \in U^+$ were arbitrary, the final claim follows. □

Proposition 65 (The associated representation is a Linear RNN)**.** *Let S be a parametric RNC, and let*

$$\mathsf{S} := \langle \mathsf{C}, x^{\mathrm{init}} \rangle \blacktriangleright h$$

be a parametric analytic output extension presentation of S. Suppose that the parametric recurrent cascade presentation C is of the form

$$\mathsf{C} = g_1 \blacktriangleright \mathsf{L}_1 \blacktriangleleft_{g_2} \mathsf{L}_2 \blacktriangleleft_{g_3} \cdots \blacktriangleleft_{g_\ell} \mathsf{L}_\ell \qquad \textit{where} \qquad \mathsf{L}_i = D_{i,1} \parallel \cdots \parallel D_{i,m_i} \quad \forall i \in [1..\ell]$$

with each connecting function g_i satisfying $g_i \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}$, and each unit $D_{i,j}$ being a Parametric General Linear Recurrent Unit. Let

$$\tilde{\mathsf{S}} := \langle \tilde{\mathsf{C}}, \tilde{x}^{\mathrm{init}} \rangle \blacktriangleright \tilde{h}$$

be the Linear RNN presentation associated with S according to Definition 268.

Then, the Linear RNN representation $[\![\tilde{\mathsf{S}}]\!]$ is a Parametric Linear RNN. End of Proposition 65.

Proof. By Definition 267, for every $i \in [1..\ell]$ and $j \in [1..m_i]$, the canonical Linear Recurrent Unit representation $\tilde{D}_{i,j}$ of $D_{i,j}$ is a Parametric Linear Recurrent Unit. Hence

$$\tilde{D}_{i,j} \in \mathcal{D}^{\mathrm{p}}[\mathrm{Linear}] \qquad \forall i \in [1..\ell], \quad \forall j \in [1..m_i].$$

Therefore, for every $i \in [1..\ell]$, the flattened layer presentation

$$\tilde{\mathsf{L}}_i = \tilde{D}_{i,1} \parallel \cdots \parallel \tilde{D}_{i,m_i}$$

is a parametric recurrent layer presentation of type $\mathcal{D}^{\mathrm{p}}[\mathrm{Linear}]$.

We next consider the connecting functions. By assumption,

$$g_i \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}} \qquad \forall i \in [1..\ell].$$

In particular, the first connecting function g_1 of $\tilde{\mathsf{C}}$ is parametric analytic.

For every $i \in [2..\ell]$, Definition 268 gives

$$\tilde{g}_i\Big(\tilde{x}_{\mathrm{pre}}, u, \tilde{x}_{\mathrm{cur}};\ \theta_{g_i}\Big) = g_i\Big(\phi_{<i}^{-1}(\tilde{x}_{\mathrm{pre}}), u, \phi_{<i}^{-1}(\tilde{x}_{\mathrm{cur}});\ \theta_{g_i}\Big).$$

By Fact 59, $\phi_{<i}^{-1}$ is a linear map. Hence it is analytic. Consequently, $\tilde{g}_i$ is obtained from the parametric analytic map g_i by composition with analytic coordinate maps and linear maps. By closure of parametric analytic Euclidean maps under finite tupling and composition,

$$\tilde{g}_i \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}} \qquad \forall i \in [2..\ell].$$

It follows that

$$\tilde{\mathsf{C}} = g_1 \blacktriangleright \tilde{\mathsf{L}}_1 \blacktriangleleft_{\tilde{g}_2} \tilde{\mathsf{L}}_2 \blacktriangleleft_{\tilde{g}_3} \cdots \blacktriangleleft_{\tilde{g}_\ell} \tilde{\mathsf{L}}_\ell$$

is a parametric recurrent $\mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}$-cascade $\mathcal{D}^{\mathrm{p}}[\mathrm{Linear}]$-presentation.

We finally consider the output function. Since S is a parametric analytic output extension presentation,

$$h \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}.$$

By Definition 268,

$$\tilde{h}\Big(\tilde{x}_{\mathrm{pre}}, u, \tilde{x}_{\mathrm{cur}};\ \theta_h\Big) = h\Big(\phi_{\mathsf{C}}^{-1}(\tilde{x}_{\mathrm{pre}}), u, \phi_{\mathsf{C}}^{-1}(\tilde{x}_{\mathrm{cur}});\ \theta_h\Big).$$

By Fact 59, ϕ_{C}^{-1} is linear and hence analytic. Thus, again by closure under finite tupling and composition,

$$\tilde{h} \in \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}}.$$

Therefore,

$$\tilde{\mathsf{S}} = \langle \tilde{\mathsf{C}}, \tilde{x}^{\mathrm{init}} \rangle \blacktriangleright \tilde{h}$$

is a parametric analytic output extension presentation of parametric recurrent cascade having type $(\mathcal{D}^{\mathrm{p}}[\mathrm{Linear}], \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}})$. Consequently,

$$\tilde{S} = [\![\tilde{\mathsf{S}}]\!]$$

is a $\mathcal{D}^{\mathrm{p}}[\mathrm{Linear}]$-RNC, that is, a Parametric Linear RNC, and hence a Parametric Linear RNN. □

Remark 48 (Dependence on the chosen RNC presentation). The canonical Linear Recurrent Unit representation of Definition 267 is determined by the fixed flattening convention.

At the level of a complete RNC, however, the construction of Definition 268 is relative to a chosen output extension presentation S, and in particular to its underlying recurrent cascade presentation C. Indeed, the presentation C determines the layer and unit decomposition, the corresponding product decomposition and ordering of the state space, and therefore the presentation-induced state flattening ϕ_{C}. It consequently also determines the flattened connecting functions, flattened output function, and the resulting Linear RNN presentation $\tilde{\mathsf{S}}$.

Thus, different presentations of the same parametric RNC may induce different Linear RNN representations. The construction is therefore canonical relative to the chosen presentation, but need not be canonical as a function of the presented parametric system alone. *End of Remark 48.*

R.3 Contractive Linear RNNs

Definition 269. Euclidean dynamics $D = \langle X, U, f \rangle$ are:

- *contractive* if, for every $u \in U$, there exists $\alpha_u \in [0, 1)$ such that
$$\|f(x, u) - f(x', u)\| \leq \alpha_u \cdot \|x - x'\| \qquad \forall\, x \in X,\ \forall\, x' \in X;$$
- *contractive on bounded input sets (b-contractive)* if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $\alpha_r \in [0, 1)$ such that
$$\|f(x, u) - f(x', u)\| \leq \alpha_r \cdot \|x - x'\| \qquad \forall\, u \in U_{\leq r},\ \forall\, x \in X,\ \forall\, x' \in X\,;$$
- *uniformly contractive (u-contractive)* if there exists $\alpha \in [0, 1)$ such that
$$\|f(x, u) - f(x', u)\| \leq \alpha \cdot \|x - x'\| \qquad \forall\, u \in U,\ \forall\, x \in X,\ \forall\, x' \in X\,.$$

Then,

- a class $\mathcal{D}$ of dynamics is *contractive*, *b-contractive*, or *u-contractive* if all dynamics $D \in \mathcal{D}$ are contractive, b-contractive, and u-contractive, respectively;
- parametric dynamics D are *contractive*, *b-contractive*, or *u-contractive* if all instances $D[\theta]$ are contractive, b-contractive, and u-contractive, respectively;
- a system S is *contractive*, *b-contractive*, or *u-contractive* if its dynamics $\mathrm{dyn}(S)$ are contractive, b-contractive, and u-contractive, respectively;
- a parametric system S is *contractive*, *b-contractive*, or *u-contractive* if all instances $S[\theta]$ are contractive, b-contractive, and u-contractive, respectively;
- a class $\mathcal{S}$ of systems is *contractive*, *b-contractive*, or *u-contractive* if every system $S \in \mathcal{S}$ is contractive, b-contractive, and u-contractive, respectively.

End of Definition 269.

Proposition 66. *Let $D = \langle X, U, f \mid A, B \rangle$ be a Linear Recurrent Unit, with $X \subseteq \mathbb{R}^{d_{\mathrm{state}}}$. Let us define*
$$\mathrm{Lip}_X(M) := \sup_{\substack{x, x' \in X \\ x \neq x'}} \frac{\|M \cdot x - M \cdot x'\|}{\|x - x'\|} \qquad \text{for } M \in \mathbb{R}^{d_{\mathrm{state}} \times d_{\mathrm{state}}}.$$

Then:

(1) D is contractive if and only if
$$\mathrm{Lip}_X(A(u)) < 1 \qquad \forall u \in U;$$

(2) D is b-contractive if and only if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $\alpha_r \in [0, 1)$ such that
$$\sup_{u \in U_{\leq r}} \mathrm{Lip}_X(A(u)) \leq \alpha_r;$$

(3) D is u-contractive if and only if there exists $\alpha \in [0,1)$ such that

$$\sup_{u \in U} \mathrm{Lip}_X(A(u)) \leq \alpha.$$

Proof. For every $x, x' \in X$ and $u \in U$,

$$\begin{aligned} f(x,u) - f(x',u) &= A(u) \cdot x + B(u) - A(u) \cdot x' - B(u) \\ &= A(u) \cdot x - A(u) \cdot x'. \end{aligned}$$

Hence,

$$\sup_{\substack{x,x' \in X \\ x \neq x'}} \frac{\|f(x,u) - f(x',u)\|}{\|x - x'\|} \;=\; \sup_{\substack{x,x' \in X \\ x \neq x'}} \frac{\|A(u) \cdot x - A(u) \cdot x'\|}{\|x - x'\|} \;=\; \mathrm{Lip}_X(A(u)).$$

Thus, for every $u \in U$, $\mathrm{Lip}_X(A(u))$ is precisely the Lipschitz constant of the state map $f(\cdot, u)$ on X.

Therefore, D is contractive if and only if

$$\mathrm{Lip}_X(A(u)) < 1 \qquad\qquad \forall u \in U,$$

proving the first claim.

Similarly, D is b-contractive if and only if, for every $r \in \mathbb{R}^{\geq 0}$, the Lipschitz constants of the state maps $f(\cdot, u)$ admit a common bound $\alpha_r < 1$ over $u \in U_{\leq r}$. By the equality above, this is equivalent to the existence of $\alpha_r \in [0,1)$ such that

$$\sup_{u \in U_{\leq r}} \mathrm{Lip}_X(A(u)) \leq \alpha_r.$$

Finally, D is u-contractive if and only if the Lipschitz constants of the state maps admit a common bound $\alpha < 1$ over all $u \in U$. Again, by the equality above, this is equivalent to

$$\sup_{u \in U} \mathrm{Lip}_X(A(u)) \leq \alpha$$

for some $\alpha \in [0,1)$. □

Proposition 67. *A Linear Recurrent Unit $D = \langle X, U, f \mid A, B \rangle$ having a full-Euclidean state space $X = \mathbb{R}^{d_{\text{state}}}$ is:*

- *contractive if and only if*
$$\|A(u)\| < 1 \qquad\qquad \forall u \in U;$$
- *b-contractive if, for every $r \in \mathbb{R}^{\geq 0}$, there exists $\alpha_r \in [0,1)$ such that*
$$\sup_{u \in U_{\leq r}} \|A(u)\| \leq \alpha_r;$$
- *u-contractive if there exists $\alpha \in [0,1)$ such that*
$$\sup_{u \in U} \|A(u)\| \leq \alpha.$$

Proof. Since D is a Linear Recurrent Unit,

$$f(x,u) = A(u) \cdot x + B(u).$$

Therefore, for every $x, x' \in X$ and every $u \in U$,

$$\begin{aligned} \|f(x,u) - f(x',u)\| &= \|A(u) \cdot x + B(u) - A(u) \cdot x' - B(u)\| \\ &= \|A(u) \cdot (x - x')\|. \end{aligned}$$

Since $X = \mathbb{R}^{d_{\text{state}}}$, every nonzero vector $v \in \mathbb{R}^{d_{\text{state}}}$ can be written as $v = x - x'$ for some $x, x' \in X$. Hence, by the definition of the induced operator norm,

$$\begin{aligned}\sup_{\substack{x,x' \in X \\ x \neq x'}} \frac{\|f(x,u) - f(x',u)\|}{\|x - x'\|} &= \sup_{\substack{x,x' \in X \\ x \neq x'}} \frac{\|A(u) \cdot (x - x')\|}{\|x - x'\|} \\ &= \sup_{v \in \mathbb{R}^{d_{\text{state}}} \setminus \{\mathbf{0}\}} \frac{\|A(u) \cdot v\|}{\|v\|} \\ &= \|A(u)\|.\end{aligned}$$

Thus, for every $u \in U$, the Lipschitz constant of the state map $f(\cdot, u)$ is exactly $\|A(u)\|$. The three claims then follow directly from the definitions of contractivity, b-contractivity, and u-contractivity. □

Definition 270 (Contractive Linear RNNs)**.** A Linear RNN is *(unitwise-)contractive*, *(unitwise-)b-contractive*, or *(unitwise-)u-contractive* if all its units are contractive, b-contractive, and u-contractive, respectively. *End of Definition 270.*

R.4 Smoothness up to the Closure

Definition 271 (Closure)**.** Let X be a metric space having ambient space Ω_X. The *closure* of X in (Ω_X), denoted by $\mathrm{CL}_{\Omega_X}(X)$, is the set of all points $x \in \Omega_X$ such that every open ball centred at x intersects X; namely,

$$\mathrm{CL}_{\Omega_X}(X) := \left\{ x \in X \mid \forall \epsilon \in \mathbb{R}^{>0},\ \mathrm{Ball}_{\Omega_X}(x, \epsilon) \cap X \neq \emptyset \right\}.$$

Equivalently, $\mathrm{CL}_{\Omega_X}(X)$ is the smallest closed subset of Ω_X containing X.

When the ambient space Ω_X is clear from the context, we simply write $\mathrm{CL}(X)$ in place of $\mathrm{CL}_{\Omega_X}(X)$. *End of Definition 271.*

Definition 272 (Continuous and differentiable extension to the closure)**.** Let $Z \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^q$.

A function $g : Z \to Y$ is:

- *continuous up to the closure of its domain* if there exists a continuous function $\widetilde{g} : \mathrm{CL}(Z) \to \mathbb{R}^q$ such that $\widetilde{g}|_Z = g$;
- *continuously-differentiable up to the closure of its domain* if there exist an open set $O \subseteq \mathbb{R}^d$ with $\mathrm{CL}(Z) \subseteq O$ and a continuously-differentiable function $\widetilde{g} : O \to \mathbb{R}^q$ such that $\widetilde{g}|_Z = g$.

End of Definition 272.

Lemma 33 (Boundedness on bounded subsets)**.** *Every function between finite-dimensional Euclidean spaces that is continuous up to the closure of its domain is boundedness-preserving.*

Proof. Let $g : Z \to Y$ be continuous up to the closure of its domain, and let $K \subseteq Z$ be bounded.

The closure $\mathrm{CL}(K)$ is closed and bounded in $\mathbb{R}^d$, and hence compact. Also, it holds that $\mathrm{CL}(K) \subseteq \mathrm{CL}(Z)$.

Let $\widetilde{g} : \mathrm{CL}(Z) \to \mathbb{R}^q$ be a continuous extension of g. Then $\widetilde{g}(\mathrm{CL}(K))$ is compact and therefore bounded.

Since $g(K) \subseteq \widetilde{g}(\mathrm{CL}(K))$, we conclude that $g(K)$ is bounded. □

Lemma 34 (Bounded derivatives on bounded subsets)**.** *Let $g : Z \to Y$ be continuously-differentiable up to the closure of its domain. Then, for every bounded $K \subseteq Z$, there exists $M_K \in \mathbb{R}^{\geq 0}$ such that*

$$\|\mathsf{D}g(z)\| \leq M_K \qquad \forall z \in K.$$

Proof. Let $\widetilde{g} : O \to \mathbb{R}^q$ be a continuously-differentiable extension of g to an open neighbourhood O of $\mathrm{CL}(Z)$.

Since K is bounded, $\mathrm{CL}(K)$ is compact. The map

$$z \mapsto \|\mathsf{D}\widetilde{g}(z)\|$$

is continuous and therefore attains a finite maximum on $\mathrm{CL}(K)$. The claim follows by restriction to K. □

Lemma 35 (Uniform strictness on bounded subsets)**.** *Let $A : Z \to \mathbb{R}^{d\times d}$ be continuous up to the closure of its domain, and let $\widetilde{A} : \mathrm{CL}(Z) \to \mathbb{R}^{d\times d}$ be a continuous extension satisfying*

$$\|\widetilde{A}(z)\| < 1 \qquad\qquad \forall z \in \mathrm{CL}(Z).$$

Then, for every bounded set $K \subseteq Z$, there exists $\alpha_K \in [0, 1)$ such that

$$\|A(z)\| \leq \alpha_K \qquad\qquad \forall z \in K.$$

Proof. Since K is bounded, we have that $\mathrm{CL}(K)$ is compact and $\mathrm{CL}(K) \subseteq \mathrm{CL}(Z)$.

The function $z \mapsto \|\widetilde{A}(z)\|$ is continuous and therefore attains its maximum on $\mathrm{CL}(K)$. Let us define

$$\alpha_K := \max_{z\in\mathrm{CL}(K)} \|\widetilde{A}(z)\|.$$

By hypothesis, every value of $\|\widetilde{A}(z)\|$ on $\mathrm{CL}(K)$ is strictly smaller than one. It follows that $\alpha_K < 1$, and hence the claim. □

R.4.1 Smooth Unitwise-contractive Linear RNNs

Definition 273 (Smooth unitwise-contractive Linear RNNs)**.** A parametric Linear RNN S is *smooth (unitwise-)contractive* if the following conditions hold:

- for every constituent parametric Linear Recurrent Unit

  $$D_{i,j} = \langle X_{i,j}, U_i, f_{i,j}; \Theta^f_{i,j} \mid A_{i,j}, B_{i,j}\rangle,$$

  with parametric input functions

  $$A_{i,j} : U_i \times \Theta^f_{i,j} \to \mathbb{R}^{d_{i,j}\times d_{i,j}}, B_{i,j} : U_i \times \Theta^f_{i,j} \to \mathbb{R}^{d_{i,j}},$$

  the following two conditions hold jointly:

  (1) the input functions $A_{i,j}$ and $B_{i,j}$ are continuously-differentiable up to the closures of their domains;
  (2) the parametric input function $A_{i,j}$ admits an extension

  $$\widetilde{A}_{i,j} : \mathrm{CL}(U_i \times \Theta^f_{i,j}) \to \mathbb{R}^{d_{i,j}\times d_{i,j}}$$

  satisfying

  $$\|\widetilde{A}_{i,j}(v, \theta)\| < 1 \qquad\qquad \forall (v, \theta) \in \mathrm{CL}(U_i \times \Theta^f_{i,j}); \tag{182}$$

- every layer's output function h_i is continuously-differentiable up to the closure of its domain.

End of Definition 273.

Remark 49. Condition (182) implies that every constituent Linear Recurrent Unit is contractive. Indeed, for every $x, x' \in X_{i,j}$, $u \in U_i$, and $\theta \in \Theta^f_{i,j}$,

$$\begin{aligned}\|f_{i,j}(x, u; \theta) - f_{i,j}(x', u; \theta)\| &= \|A_{i,j}(u; \theta) \cdot (x - x')\| \\ &\leq \|A_{i,j}(u; \theta)\| \cdot \|x - x'\| \\ &< \|x - x'\|.\end{aligned}$$

End of Remark 49.

Remark 50 (Why contraction on the closure is required). It is not sufficient to assume only that

$$\|A(u)\| < 1 \qquad \forall u \in U$$

together with continuous extendability.

For example, for

$$U = (0,1) \qquad \text{and} \qquad A(u) = u,$$

A extends continuously to $[0,1]$ and satisfies $|A(u)| < 1$ for every $u \in U$, but

$$\sup_{u \in U} |A(u)| = 1.$$

Requiring strict contraction also on the closure rules out this boundary degeneration. *End of Remark 50.*

Lemma 36 (Bounded reachability of smooth contractive Linear RNNs)**.** *Let S be a smooth contractive parametric Linear RNN.*

Let

$$K_X \subseteq X, \qquad K_U \subseteq U, \qquad K_\Theta \subseteq \Theta$$

be bounded sets.

Then there exists $R \in \mathbb{R}^{\geq 0}$ such that, for every

$$x_0 \in K_X, \qquad \theta \in K_\Theta, \qquad u \in K_U^\omega,$$

the corresponding state sequence satisfies

$$\|x_t^{x_0,u,\theta}\| \leq R \qquad \forall t \in \mathbb{N}.$$

Furthermore, the state, input, and output sequences of every constituent layer are uniformly bounded over these choices. End of Lemma 36.

Proof. Let

$$\mathrm{dyn}(S) = S_1 \ltimes \cdots \ltimes S_m$$

be the cascade decomposition of the dynamics of S. We proceed inductively over its layers.

Consider first layer S_1. Its input set K_U is bounded.

For every constituent Linear Recurrent Unit $D_{1,j}$, let

$$K_{\Theta_{1,j}^f}$$

denote the projection of K_Θ onto $\Theta_{1,j}^f$. This set is bounded.

Hence

$$K_U \times K_{\Theta_{1,j}^f}$$

is bounded. By Lemma 35, there exists $\alpha_{1,j} \in [0,1)$ such that

$$\|A_{1,j}(v;\theta)\| \leq \alpha_{1,j}$$

for every

$$v \in K_U, \qquad \theta \in K_{\Theta_{1,j}^f}.$$

Moreover, by Lemma 33, there exists $\beta_{1,j} \in \mathbb{R}^{\geq 0}$ such that

$$\|B_{1,j}(v;\theta)\| \leq \beta_{1,j}$$

on the same bounded set.

Since K_X is bounded, the initial-state component associated with $D_{1,j}$ is uniformly bounded over $x_0 \in K_X$. Thus there exists $R_{1,j}^{\mathrm{init}} \in \mathbb{R}^{\geq 0}$ such that

$$\|x_{1,j,0}\| \leq R_{1,j}^{\mathrm{init}}.$$

Therefore,

$$\begin{aligned}\|x_{1,j,t}\| &\leq \alpha_{1,j} \cdot \|x_{1,j,t-1}\| + \beta_{1,j} \\ &\leq R_{1,j}^{\text{init}} + \frac{\beta_{1,j}}{1-\alpha_{1,j}} \qquad \forall t \in \mathbb{N}.\end{aligned}$$

Hence the state sequence of every unit of the first layer is uniformly bounded. Since the number of units is finite, the state sequence of S_1 is uniformly bounded.

The arguments supplied to the output function h_1 consist of:

- the previous state of S_1;
- the input to S_1;
- the updated state of S_1;
- the corresponding output-function parameters.

All these quantities range over bounded sets.

Since h_1 is continuously-differentiable up to the closure of its domain, it is in particular continuous up to the closure. Hence, by Lemma 33, the output sequence of S_1 is uniformly bounded.

Now let $i \in [2..m]$ and suppose inductively that the output sequence of S_{i-1} is uniformly bounded. By the definition of cascade composition, this output sequence is the input sequence of S_i. Therefore the input sequence of S_i ranges over a bounded set.

Repeating the preceding argument for every unit $D_{i,j}$ gives:

- a uniform contraction constant $\alpha_{i,j} < 1$ on the bounded set of reachable layer inputs and parameters;
- a uniform bound $\beta_{i,j}$ on the corresponding additive terms;
- consequently, a uniform bound on every unit state;
- consequently, a uniform bound on the state of S_i;
- by closure-continuity of h_i, a uniform bound on its output.

Thus the induction holds for every layer.

Since the complete state of the Linear RNN is the finite tuple of the states of its layers, there exists $R \in \mathbb{R}^{\geq 0}$ such that

$$\|x_t^{x_0,u,\theta}\| \leq R$$

uniformly over $x_0 \in K_X$, $\theta \in K_\Theta$, $u \in K_U^\omega$, and $t \in \mathbb{N}$. □

Lemma 37 (State-transition decay on bounded sets)**.** *Let S be a smooth contractive Linear RNN.*

For every bounded

$$K_X \subseteq X, \qquad K_U \subseteq U, \qquad K_\Theta \subseteq \Theta,$$

there exist constants

$$C \in \mathbb{R}^{>0}, \qquad \rho \in [0,1)$$

such that

$$\left\| J_t^{x_0,u,\theta} \cdot J_{t-1}^{x_0,u,\theta} \cdots J_{s+1}^{x_0,u,\theta} \right\| \leq C \cdot \rho^{t-s}$$

for every

$$x_0 \in K_X, \qquad \theta \in K_\Theta, \qquad u \in K_U^\omega,$$

and every $s, t \in \mathbb{N}$ satisfying $s < t$. End of Lemma 37.

Proof. By Lemma 36, all layer states and layer inputs reachable from the considered bounded sets are uniformly bounded.

For every unit $D_{i,j}$, its input and parameter arguments therefore range over a bounded set. By Lemma 35, there exists $\alpha_{i,j} < 1$ such that

$$\|A_{i,j}(v;\theta)\| \le \alpha_{i,j}$$

uniformly over all reachable arguments.

Since there are finitely many units, we have

$$\alpha := \max_{\substack{i\in[1..m]\\ j\in[1..n_i]}} \alpha_{i,j},$$

satisfying $\alpha < 1$.

The one-step state Jacobian is block lower triangular with respect to the cascade layers:

$$J_t = \begin{pmatrix} A_{1,t} & 0 & \cdots & 0 \\ C_{2,1,t} & A_{2,t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1,t} & C_{m,2,t} & \cdots & A_{m,t} \end{pmatrix}.$$

Every diagonal block $A_{i,t}$ is block diagonal with blocks given by the state-update matrices of the units in layer i. Hence

$$\|A_{i,t}\| \le \alpha \qquad \forall i, t.$$

The off-diagonal blocks $C_{i,j,t}$ are finite combinations of:

- values and derivatives of the functions $A_{i,j}$ and $B_{i,j}$;
- values and derivatives of the output functions h_i;
- reachable state and input values.

All their arguments range over bounded sets. Since all the relevant functions are continuously-differentiable up to the closures of their domains, Lemma 34 and Lemma 33 imply that these quantities are uniformly bounded.

Since the architecture is finite, there exists $M \in \mathbb{R}^{\ge 0}$ such that

$$\|C_{i,j,t}\| \le M \qquad \forall j < i,\ \forall t.$$

Fix any

$$\rho \in (\alpha, 1).$$

Let h be an arbitrary state perturbation and define

$$z_\ell := J_{s+\ell} \cdots J_{s+1} \cdot h, \qquad z_0 := h.$$

Write

$$z_\ell = \langle z_\ell^{(1)}, \ldots, z_\ell^{(m)} \rangle$$

according to the cascade layers.

For the first layer,

$$z_\ell^{(1)} = A_{1,s+\ell} \cdot z_{\ell-1}^{(1)},$$

and hence

$$\|z_\ell^{(1)}\| \le \alpha^\ell \|h\| \le \rho^\ell \|h\|.$$

Suppose inductively that, for every $j < i$, there exists $K_j \in \mathbb{R}^{>0}$ such that

$$\|z_\ell^{(j)}\| \le K_j \rho^\ell \|h\|.$$

Then

$$z_\ell^{(i)} = A_{i,s+\ell} z_{\ell-1}^{(i)} + \sum_{j<i} C_{i,j,s+\ell} z_{\ell-1}^{(j)}.$$

Consequently,

$$\|z_\ell^{(i)}\| \le \alpha \|z_{\ell-1}^{(i)}\| + M \sum_{j<i} K_j \rho^{\ell-1} \|h\|.$$

Iterating gives

$$\begin{aligned}
\|z_\ell^{(i)}\| &\le \alpha^\ell \|h\| + M \sum_{j<i} K_j \sum_{q=1}^{\ell} \alpha^{\ell-q} \rho^{q-1} \|h\| \\
&= \alpha^\ell \|h\| + M \sum_{j<i} K_j \frac{\rho^\ell - \alpha^\ell}{\rho - \alpha} \|h\| \\
&\le \left(1 + \frac{M \sum_{j<i} K_j}{\rho - \alpha}\right) \rho^\ell \|h\|.
\end{aligned}$$

Thus the induction holds for

$$K_i := 1 + \frac{M \sum_{j<i} K_j}{\rho - \alpha}.$$

Since the number of layers is finite,

$$C := \sqrt{\sum_{i=1}^{m} K_i^2}$$

is finite and

$$\|z_\ell\| \le C \rho^\ell \|h\|.$$

Taking $\ell = t - s$ and then the supremum over $h \neq 0$ proves the claim. □

R.5 Positive-Margin Expressivity of Smooth Contractive Linear RNNs

The next lemma isolates the analytic property used by the expressivity proof. For a finite input alphabet, smooth unit-wise contractivity (Definition 273) implies that two computations processing the same sufficiently long suffix forget their possibly different preceding prefixes uniformly.

Lemma 38 (Uniform suffix forgetting)**.** *Let S be an instance of a smooth contractive Linear RNN in the sense of Definition 273, and let* $\mathrm{enc} : \Sigma \to U$ *be an encoder from a finite alphabet Σ. Then, for some sequence*

$$(\varepsilon_n)_{n \in \mathbb{N}^{>0}} \in \mathbb{R}^{\ge 0 \omega} \qquad \textit{with} \qquad \varepsilon_n \to 0 \quad \textit{as} \quad n \to \infty,$$

we have:

$$d_Y\Big(S\big(\mathrm{enc}(vz)\big),\, S\big(\mathrm{enc}(wz)\big)\Big) \;\le\; \varepsilon_n \qquad \forall n \in \mathbb{N}^{>0},\ \forall v, w \in \Sigma^*,\ \forall z \in \Sigma^n. \tag{183}$$

End of Lemma 38.

Proof. Since Σ is finite, the encoded input set $\mathrm{enc}(\Sigma)$ is bounded. Applying Lemma 36 with the fixed initial states and fixed parameter values of S shows that the input, state, and output sequences of every constituent layer are uniformly bounded over all encoded words.

For every constituent recurrent unit $D_{i,j}$, the set of input-parameter pairs encountered in these computations is bounded. Hence the strict contraction-on-the-closure requirement of Definition 273, together with Lemma 35, yields a constant

$$\alpha_{i,j} \in [0, 1)$$

satisfying

$$\|A_{i,j}(u)\| \le \alpha_{i,j} \tag{184}$$

for every layer input u occurring while processing encoded words. Furthermore, continuous differentiability up to the closure implies that each coefficient function $A_{i,j}$ and $B_{i,j}$, and each layer output function h_i, is uniformly continuous on the closure of every bounded set of arguments encountered in these computations.

Consider now two computations that have processed arbitrary prefixes $v, w \in \Sigma^*$ and subsequently process the same suffix

$$z = z_1 \cdots z_n.$$

We prove inductively over the layers that the corresponding layer states and layer outputs converge uniformly to one another as the length of the common suffix increases.

Consider first a recurrent unit $D_{1,j}$ in the first layer. During the common suffix the two computations receive exactly the same input $\mathrm{enc}(z_t)$ at every time t. If x_t and x'_t denote the corresponding unit states after the first t symbols of the common suffix, then Equation (184) gives

$$\|x_t - x'_t\| \;\le\; \alpha_{1,j} \cdot \|x_{t-1} - x'_{t-1}\| \;\le\; \alpha_{1,j}^t \cdot \|x_0 - x'_0\|.$$

Uniform boundedness of the reachable states therefore implies that this difference converges uniformly to zero over the prefixes v, w and common suffixes. Since the first layer contains finitely many recurrent units, the same conclusion holds for its complete state. Uniform continuity of h_1 on the corresponding bounded argument set then implies uniform convergence of the first-layer outputs.

Let now $i \in [2..m]$, and suppose inductively that the outputs of layer S_{i-1} in the two computations converge uniformly to one another during an increasingly long common suffix. These outputs are the inputs u_t and u'_t to layer S_i, and therefore

$$\|u_t - u'_t\| \to 0 \tag{185}$$

uniformly over the prefixes and common suffixes as $t \to \infty$.

Consider a recurrent unit $D_{i,j}$ in this layer and write

$$\begin{aligned} x_t &= A_{i,j}(u_t) \cdot x_{t-1} + B_{i,j}(u_t), \\ x'_t &= A_{i,j}(u'_t) \cdot x'_{t-1} + B_{i,j}(u'_t). \end{aligned}$$

Subtracting gives

$$\begin{aligned} x_t - x'_t = {} & A_{i,j}(u_t) \cdot (x_{t-1} - x'_{t-1}) && (186) \\ & + \big(A_{i,j}(u_t) - A_{i,j}(u'_t)\big) \cdot x'_{t-1} && (187) \\ & + B_{i,j}(u_t) - B_{i,j}(u'_t). && (188) \end{aligned}$$

The states x'_t are uniformly bounded. Together with Equation (185) and the uniform continuity of $A_{i,j}$ and $B_{i,j}$ on the relevant bounded argument sets, Equation (186) yields a sequence $(e_t)_{t \in \mathbb{N}^{>0}}$ satisfying

$$e_t \to 0$$

uniformly over prefixes and common suffixes, such that

$$\|x_t - x'_t\| \;\le\; \alpha_{i,j} \cdot \|x_{t-1} - x'_{t-1}\| + e_t. \tag{189}$$

Iterating gives

$$\|x_t - x'_t\| \;\le\; \alpha_{i,j}^t \cdot \|x_0 - x'_0\| + \sum_{s=1}^{t} \alpha_{i,j}^{t-s} \cdot e_s.$$

The first term converges uniformly to zero by boundedness of the states. Since $\alpha_{i,j} < 1$ and $e_s \to 0$ uniformly, the convolution term also converges uniformly to zero. Hence the unit states converge uniformly to one another. Since the layer contains finitely many units, the same holds for its full state, and uniform continuity of h_i implies uniform convergence of the layer outputs.

Proceeding inductively through the finitely many layers yields

$$\sup_{\substack{v,w \in \Sigma^* \\ z \in \Sigma^n}} d_Y\big(S(\mathrm{enc}(vz)), S(\mathrm{enc}(wz))\big) \to 0 \quad \text{as} \quad n \to \infty.$$

Defining this supremum to be ε_n proves the claim. □

Theorem 53 (Positive-margin expressivity of smooth contractive Linear RNNs)**.** *The positive-margin expressivity of smooth contractive Linear RNNs is contained in the definite languages.*

End of Theorem 53.

Proof. Let S be a smooth contractive Linear RNN that recognises a language L over a finite alphabet Σ with positive margin through an encoder enc and decoder dec. Let $\gamma \in \mathbb{R}^{>0}$ be a corresponding decoder margin.

By Lemma 38, choose $N \in \mathbb{N}^{>0}$ such that

$$\varepsilon_N < \gamma.$$

Then, for every $v, w \in \Sigma^*$ and $z \in \Sigma^N$,

$$d_Y\big(S(\mathrm{enc}(vz)), S(\mathrm{enc}(wz))\big) < \gamma.$$

By the decoder-margin condition,

$$\mathrm{dec}\big(S(\mathrm{enc}(vz))\big) = \mathrm{dec}\big(S(\mathrm{enc}(wz))\big),$$

and hence

$$vz \in L \quad \Longleftrightarrow \quad wz \in L.$$

Lemma 4 therefore implies that L is definite. □

R.6 BIBS of Smooth Contractive Linear RNNs

Theorem 54. *Every smooth contractive Linear RNN satisfies BIBS.* End of Theorem 54.

Proof. Let us consider a smooth contractive Linear RNN

$$S = \langle X, U, f, x^{\mathrm{init}}, Y, h; \Theta^f, \Theta^h \rangle.$$

Let us fix arbitrary

$$\theta \in \Theta, \qquad x_0 \in X, \qquad u = (u_t)_{t \in \mathbb{T}} \in U^{\omega \mathrm{b}}.$$

Let

$$K_X := \{x_0\}, \qquad K_\Theta := \{\theta\}, \qquad K_U := \{u_t \mid t \in \mathbb{T}\}.$$

These sets are bounded.

Hence, by Lemma 36, there exists $B \in \mathbb{R}^{\geq 0}$ such that

$$\big\|x_t^{x_0,u,\theta}\big\| \leq B \qquad \forall t \in \mathbb{N}.$$

Therefore

$$\sup_{t \in \mathbb{T}} \big\|x_t^{x_0,u,\theta}\big\| \leq B.$$

Since θ, x_0, and u were arbitrary, S satisfies BIBS. □

R.7 BIBSD and nnVSD of Smooth Contractive Linear RNNs

Theorem 55. *Every smooth contractive Linear RNN satisfies BIBSD.* End of Theorem 55.

Proof. Let $r \in \mathbb{R}^{\geq 0}$, let $x_0 \in X_{\leq r}$, and let $u = (u_t)_{t \in \mathbb{T}} \in U^\omega_{\leq r}$ be arbitrary.

Since the parametric input functions $A_{i,j}$ and $B_{i,j}$ of every Linear Recurrent Unit and the output functions h_i of every Linear Recurrent Layer are continuously-differentiable up to the closures of their domains, they are differentiable at every point of their original domains. Therefore, by finite composition and repeated application of the chain rule, for every $t \in \mathbb{T}$, the parameter-to-state function χ_t^{u,x_0} is differentiable at every $\theta \in \Theta_{\leq r}$. This establishes the differentiability condition in the definition of BIBSD.

For every $\theta \in \Theta_{\leq r}$, let

$$x_t^\theta := \begin{cases} x_0 & \text{if } t = 0, \\ f(x_{t-1}^\theta, u_t; \theta) & \text{otherwise,} \end{cases}$$

and define the one-step state and parameter Jacobians

$$J_t^\theta := \mathsf{D} f_{u_t}^\theta\big(x_{t-1}^\theta\big), \qquad G_t^\theta := \mathsf{D} f_{u_t}^{x_{t-1}^\theta}(\theta),$$

where

$$f_{u_t}^\theta(x) := f(x, u_t; \theta), \qquad f_{u_t}^x(\vartheta) := f(x, u_t; \vartheta).$$

By Lemma 37, there exist

$$C_r \in \mathbb{R}^{>0}, \qquad \rho_r \in [0, 1),$$

which we may take with $C_r \geq 1$, such that

$$\big\| J_t^\theta \cdot J_{t-1}^\theta \cdots J_{s+1}^\theta \big\| \leq C_r \cdot \rho_r^{t-s} \tag{190}$$

for every $\theta \in \Theta_{\leq r}$ and every $s, t \in \mathbb{N}$ satisfying $s < t$.

We next uniformly bound the one-step parameter Jacobians. By Lemma 36, there exists $R_r \in \mathbb{R}^{\geq 0}$ such that

$$\|x_t^\theta\| \leq R_r \qquad \forall\, \theta \in \Theta_{\leq r},\ \forall\, u \in U_{\leq r}^\omega,\ \forall\, x_0 \in X_{\leq r},\ \forall\, t \in \mathbb{N}.$$

Thus all states, inputs, and parameters occurring in the one-step parameter Jacobians range over bounded sets. Each block of G_t^θ is obtained, by the chain rule, from finitely many values and derivatives of the parametric input functions $A_{i,j}$ and $B_{i,j}$ and of the layer output functions h_i. Since all their arguments range over bounded sets and these functions are continuously-differentiable up to the closures of their domains, Lemmas 33 and 34 imply that all such values and derivatives are uniformly bounded.

Since the architecture contains finitely many layers, units, and parameters, there exists

$$M_r \in \mathbb{R}^{\geq 0}$$

such that

$$\|G_t^\theta\| \leq M_r \qquad \forall (\theta, t) \in \Theta_{\leq r} \times \mathbb{T}. \tag{191}$$

For every $\theta \in \Theta_{\leq r}$ and $t \in \mathbb{N}$, define the parameter sensitivity

$$Z_t^\theta := \mathsf{D}\chi_t^{u, x_0}(\theta).$$

Since the initial state x_0 is independent of the parameters,

$$Z_0^\theta = 0 \qquad \forall \theta \in \Theta_{\leq r}.$$

By the chain rule,

$$Z_t^\theta \;=\; J_t^\theta \cdot Z_{t-1}^\theta + G_t^\theta \qquad \forall (\theta, t) \in \Theta_{\leq r} \times \mathbb{T}.$$

Iterating this recurrence gives

$$Z_t^\theta \;=\; G_t^\theta + \sum_{k=1}^{t-1} J_t^\theta \cdot J_{t-1}^\theta \cdots J_{k+1}^\theta \cdot G_k^\theta.$$

Hence, by Equations (190) and (191), and using $C_r \geq 1$,

$$\|Z_t^\theta\| \;\leq\; M_r + C_r \cdot M_r \sum_{k=1}^{t-1} \rho_r^{t-k} \;\leq\; C_r \cdot M_r \sum_{k=0}^{t-1} \rho_r^k \;\leq\; \frac{C_r \cdot M_r}{1 - \rho_r}.$$

Let us define

$$B_r := \frac{C_r \cdot M_r}{1 - \rho_r}.$$

Then

$$\|\mathsf{D}\chi_t^{u, x_0}(\theta)\| \leq B_r \qquad \forall\, \theta \in \Theta_{\leq r},\ \forall\, t \in \mathbb{T}.$$

Hence both conditions in the definition of BIBSD hold. Since r, x_0, and u were arbitrary, S satisfies BIBSD. □

Theorem 56 (Vanishing state derivatives of smooth contractive Linear RNNs)**.** *Let S be a smooth contractive Linear RNN having input space U and initial state x^{init}, and let Θ be the parameter space of its dynamics* $\mathrm{dyn}(S)$.

For every $\theta \in \Theta$ and every $u \in U^{\omega\mathrm{b}}$,

$$\lim_{t\to\infty} \left\| \mathsf{D}\chi_t^{u,\theta}(x^{\mathrm{init}}) \right\| = 0,$$

where $(\chi_t^{u,\theta})_{t\in\mathbb{T}}$ are the state-to-state functions of $\mathrm{dyn}(S)$.

In particular, S does not satisfy nnVSD. End of Theorem 56.

Proof. Let us consider arbitrary $\theta \in \Theta$ and $u = (u_t)_{t\in\mathbb{T}} \in U^{\omega\mathrm{b}}$.

Since u is bounded, there exists $r_u \in \mathbb{R}^{\geq 0}$ such that

$$u_t \in U_{\leq r_u} \qquad \forall t \in \mathbb{T}.$$

Furthermore, there exist $r_x, r_\theta \in \mathbb{R}^{\geq 0}$ such that

$$x^{\mathrm{init}} \in X_{\leq r_x} \qquad \text{and} \qquad \theta \in \Theta_{\leq r_\theta}.$$

Let $r := \max\{r_x, r_u, r_\theta\}$. Then

$$x^{\mathrm{init}} \in X_{\leq r}, \qquad \theta \in \Theta_{\leq r}, \qquad u \in U^{\omega}_{\leq r}.$$

Since the parametric input functions $A_{i,j}$ and $B_{i,j}$ of every Linear Recurrent Unit and the output functions h_i of every Linear Recurrent Layer are continuously-differentiable up to the closures of their domains, they are differentiable at every point of their original domains. Therefore the dynamics function f of the Linear RNN is differentiable in its state argument. Hence $\chi_t^{u,\theta}$ is differentiable at x^{init} for every $t \in \mathbb{T}$.

Let

$$J_t^{u,\theta} := \mathsf{D}f_{u_t}^{\theta}\left(x_{t-1}^{u,\theta}\right)$$

denote the one-step state Jacobian along the corresponding trajectory, where

$$f_{u_t}^{\theta}(x) := f(x, u_t; \theta) \qquad \text{and} \qquad x_t^{u,\theta} := \begin{cases} x_0 & \text{if } t = 0, \\ f_{u_t}^{\theta}(x_{t-1}^{u,\theta}) & \text{otherwise.} \end{cases}$$

By repeated application of the chain rule,

$$\mathsf{D}\chi_t^{u,\theta}(x^{\mathrm{init}}) \;=\; J_t^{u,\theta} \cdot J_{t-1}^{u,\theta} \cdots J_1^{u,\theta} \qquad \forall t \in \mathbb{T}.$$

By Lemma 37, there exist $C_r \in \mathbb{R}^{>0}$ and $\rho_r \in [0, 1)$ such that

$$\left\| J_t^{u,\theta} \cdot J_{t-1}^{u,\theta} \cdots J_1^{u,\theta} \right\| \;\leq\; C_r \cdot \rho_r^t \qquad \forall t \in \mathbb{T}.$$

Therefore,

$$\left\| \mathsf{D}\chi_t^{u,\theta}(x^{\mathrm{init}}) \right\| \;\leq\; C_r \cdot \rho_r^t \qquad \forall t \in \mathbb{T}.$$

Since $\rho_r < 1$, it follows that

$$\lim_{t\to\infty} \left\| \mathsf{D}\chi_t^{u,\theta}(x^{\mathrm{init}}) \right\| \;=\; 0.$$

Since $\theta \in \Theta$ and $u \in U^{\omega\mathrm{b}}$ are arbitrary, the state derivatives vanish for every parameter instance and every bounded input sequence.

Suppose, by contradiction, that S satisfies nnVSD. Then there exist $\theta \in \Theta$, $u \in U^{\omega\mathrm{b}}$, and $B \in \mathbb{R}^{>0}$ such that

$$\left\| \mathsf{D}\chi_t^{u,\theta}(x^{\mathrm{init}}) \right\| \geq B \qquad \forall t \in \mathbb{T}.$$

This contradicts the convergence to zero established above.

Therefore S does not satisfy nnVSD. □

S Analysis of DeltaNet

Table of Contents

S.1 Definition of DeltaNet 296
S.2 Overview of the Analysis 298
S.3 BIBS Instability of DeltaNet 298
S.4 BIBSD Instability of DeltaNet 300
S.5 BIBLG Instability of DeltaNet 303
S.6 DeltaNet: Technical Lemmas on Instability 307
S.7 DeltaNet: Deferred Proofs 308

S.1 Definition of DeltaNet

Definition 274 (Generic DeltaNet Recurrent Units)**.** Let $c \in \{1, 2\}$, let $d_{\mathrm{k}}, d_{\mathrm{v}} \in \mathbb{N}^{>0}$, and let $d_{\mathrm{kv}} := (d_{\mathrm{k}}, d_{\mathrm{v}})$.

The *admissible DeltaNet input space* of key-value dimensions d_{kv} is

$$\mathcal{U}_{\mathrm{kv}} := [0, 1] \times (0, 1) \times \mathbb{S}^{d_{\mathrm{k}}} \times \mathbb{R}^{d_{\mathrm{v}}} \quad \text{where} \quad \mathbb{S} := \{z \in \mathbb{R} : \|z\| = 1\}.$$

A *Generic DeltaNet Recurrent Unit* is a Left-Multiplicative Linear Recurrent Unit

$$D = \langle X, U, f \mid A, B \rangle \qquad X \subseteq \mathbb{R}^{d_{\mathrm{k}} \times d_{\mathrm{v}}}, \quad U \subseteq \mathcal{U}_{\mathrm{kv}},$$

where the dynamics function is as follows, for every state $H \in X$ and input $u = (g, \beta, k, v) \in U$:

$$f\big(H, (g, \beta, k, v)\big) := A(g, \beta, k) \cdot H + B(\beta, k, v)$$

with state-update matrix and additive term given respectively by

$$A(g, \beta, k) := g \cdot \big(I_{d_{\mathrm{k}}} - c \cdot \beta \cdot k \cdot k^{\top}\big) \qquad \text{and} \qquad B(\beta, k, v) := \beta \cdot k \cdot v^{\top}.$$

By expanding the definitions of A and B, the dynamics function can be equivalently written as

$$f\big(H, (g, \beta, k, v)\big) := g \cdot \big(I_{d_{\mathrm{k}}} - c \cdot \beta \cdot k \cdot k^{\top}\big) \cdot H + \beta \cdot k \cdot v^{\top}.$$

The following terminology applies:

- c is the *update hyperparameter* c,
- d_{k} is the *key dimension*,
- d_{v} is the *value dimension*,
- d_{kv} are the *key-value dimensions*.

We say that D is *non-gated* when every input $(g, \beta, k, v) \in U$ satisfies $g = 1$. *End of Definition 274.*

Definition 275 (DeltaNet and Gated DeltaNet, cf. [28, 33])**.** We first define Gated DeltaNet, and then we define DeltaNet as a special case.

Let $c \in \{1, 2\}$, let $d_{\mathrm{in}}, d_{\mathrm{k}}, d_{\mathrm{v}} \in \mathbb{N}^{>0}$, let $d_{\mathrm{kv}} := (d_{\mathrm{k}}, d_{\mathrm{v}})$, and let

$$\hat{D} = \langle X, \hat{U}, \hat{f} \mid \hat{A}, \hat{B} \rangle$$

be a Generic DeltaNet Recurrent Unit with updated hyperparameter c and key-value dimensions d_{kv}.[19]

A *Gated DeltaNet Recurrent Unit* with *update hyperparameter* c is a Parametric Left-Multiplicative Linear Recurrent Unit

$$D = \langle X, U, f; A, B \mid \Theta \rangle$$

[19] *Generic DeltaNet Recurrent Unit* are introduced in Definition 274.

having components as described next.

▸ **SubDefinition 275.1** (Input space). The input space is $U \subseteq \mathbb{R}^{d_{\text{in}}}$. *End of SD.275.1* ◂

▸ **SubDefinition 275.2** (Parameter space). The joint parameter space Θ consists of parameter values θ of the following form:

$$\theta = \big(a, (W_{\text{k}}, b_{\text{k}}), (W_{\text{v}}, b_{\text{v}}), (w_\beta, b_\beta), (w_\Delta, b_\Delta)\big) \quad \text{for} \quad \begin{cases} W_{\text{k}} \in \mathbb{R}^{d_{\text{k}} \times d_{\text{in}}}, \\ W_{\text{v}} \in \mathbb{R}^{d_{\text{v}} \times d_{\text{in}}}, \\ w_\beta \in \mathbb{R}^{d_{\text{in}}}, \\ w_\Delta \in \mathbb{R}^{d_{\text{in}}}, \end{cases} \quad \text{and} \quad \begin{cases} a \in \mathbb{R}, \\ b_{\text{k}} \in \mathbb{R}^{d_{\text{k}}}, \\ b_{\text{v}} \in \mathbb{R}^{d_{\text{v}}}, \\ b_\beta \in \mathbb{R}, \\ b_\Delta \in \mathbb{R}, \end{cases}$$

satisfying the *key-admissibility condition*

$$W_{\text{k}} \cdot u + b_{\text{k}} \;\neq\; 0 \qquad \forall u \in U. \tag{192}$$

End of SD.275.2 ◂

▸ **SubDefinition 275.3** (Recurrence input functions). The recurrence inputs are given by *parametric recurrence-input functions*

$g : U \times \Theta \to [0,1]$	$g(u;\theta) := \exp\left(-\Delta(u;\theta) \cdot \exp(a)\right)$	*decay gate*
$\beta : U \times \Theta \to (0,1)$	$\beta(u;\theta) := \sigma\big(\tilde{\beta}(u;\theta)\big)$	*write strength*
$k : U \times \Theta \to \mathbb{S}^{d_{\text{k}}-1}$	$k(u;\theta) := \tilde{k}(u;\theta) \,/\, \lVert\tilde{k}(u;\theta)\rVert$	*key*
$v : U \times \Theta \to \mathbb{R}^{d_{\text{v}}}$	$v(u;\theta) := W_{\text{v}} \cdot u + b_{\text{v}}$	*value*

where the mentioned intermediate functions are

$\Delta : U \times \Theta \to \mathbb{R}^{>0}$	$\Delta(u;\theta) := \text{softplus}\left(\tilde{\Delta}(u;\theta)\right)$	*step size*
$\tilde{\Delta} : U \times \Theta \to \mathbb{R}$	$\tilde{\Delta}(u;\theta) := w_\Delta^\top \cdot u + b_\Delta$	*step-size preactivation*
$\tilde{\beta} : U \times \Theta \to \mathbb{R}$	$\tilde{\beta}(u;\theta) := w_\beta^\top \cdot u + b_\beta$	*write-strength preactivation*
$\tilde{k} : U \times \Theta \to \mathbb{R}^{d_{\text{k}}}$	$\tilde{k}(u;\theta) := W_{\text{k}} \cdot u + b_{\text{k}}$	*key preactivation* .

End of SD.275.3 ◂

▸ **SubDefinition 275.4** (Dynamics function). The parametric dynamics function is obtained by applying the dynamics function of $\hat{D}$ to the corresponding parametrised recurrent input:

$$f(H, u; \theta) := \hat{f}\Big(H,\ (g(u;\theta), \beta(u;\theta), k(u;\theta), v(u;\theta))\Big).$$

End of SD.275.4 ◂

▸ **SubDefinition 275.5** (DeltaNet Recurrent Units with no decay gate). A *DeltaNet Recurrent Unit with no decay gate* is defined as above except that the decay gate function is the constant function $g = 1$; consequently, the unit does not need to include the parameters a, w_Δ, and b_Δ—since they are exclusive dependencies of g. *End of SD.275.5* ◂

▸ **SubDefinition 275.6** (Classes of DeltaNet Recurrent Units). We define the following classes:

- the *class of DeltaNet Recurrent Units with update hyperparameter* c is the set consisting of all DeltaNet recurrent units with update hyperparameter c, and is denoted by $\mathcal{U}[\text{DeltaNet}|c]$;
- the *class of Gated DeltaNet Recurrent Units* is the set consisting of all Gated DeltaNet recurrent units, and is denoted by $\mathcal{U}[\text{GatedDeltaNet}|c]$.

End of SD.275.6 ◂

▸ **SubDefinition 275.7** (DeltaNet Models). A *Gated DeltaNet model* with update hyperparameter c is a parametric $\mathcal{U}[\text{GatedDeltaNet}|c]$-RNC. A *DeltaNet model* with update hyperparameter c is a parametric $\mathcal{U}[\text{DeltaNet}|c]$-RNC. *End of SD.275.7* ◂

▸ **SubDefinition 275.8** (Classes of DeltaNet Models). We define the following classes:

- the *class of DeltaNet Models* with update hyperparameter c is the set consisting of each DeltaNet model with update hyperparameter c, and is denoted by $\mathcal{M}[\mathrm{DeltaNet}|c]$;
- the *class of Gated DeltaNet Models* with update hyperparameter c is the set consisting of each DeltaNet model with update hyperparameter c, and is denoted by $\mathcal{M}[\mathrm{GatedDeltaNet}|c]$;
- the *class of DeltaNet Dynamics* with update hyperparameter c is the set consisting of each $D = \mathrm{dyn}(S)$ for $S \in \mathcal{M}[\mathrm{DeltaNet}|c]$, and is denoted by $\mathcal{D}[\mathrm{DeltaNet}|c]$;
- the *class of Gated-DeltaNet Dynamics* with update hyperparameter c is the set consisting of each $D = \mathrm{dyn}(S)$ for $S \in \mathcal{M}[\mathrm{GatedDeltaNet}|c]$, and is denoted by $\mathcal{D}[\mathrm{GatedDeltaNet}|c]$.

End of SD.275.8 ◂

End of Definition 275.

Remark 51 (Decay-gate parametrisation in Gated DeltaNet). The Gated DeltaNet paper [28] specifies that the decay gate follows the Mamba2 parametrisation [15], while omitting its explicit form for brevity. The standalone official implementation of Gated DeltaNet [72] and its FLA implementation [73] make this parametrisation explicit by first forming the log-decay

$$\gamma(u) = -\exp(a) \cdot \mathrm{softplus}\left(\tilde{\Delta}(u)\right),$$

and then using its exponential as the multiplicative decay gate. Consequently, in the notation of Definition 275,

$$\begin{aligned} g(u;\theta) &= \exp\Big(-\exp(a) \cdot \mathrm{softplus}\left(\tilde{\Delta}(u;\theta)\right)\Big) \\ &= \exp\left(-\Delta(u;\theta) \cdot \exp(a)\right), \end{aligned}$$

where

$$\Delta(u;\theta) = \mathrm{softplus}\left(\tilde{\Delta}(u;\theta)\right).$$

Thus, the parametrisation used in Definition 275 is the explicit form of the Mamba2-style decay parametrisation adopted by Gated DeltaNet and used in its reference implementations. We note that the implementations store the quantity corresponding to γ as the decay variable, whereas g in Definition 275 denotes the resulting multiplicative decay gate $\exp(\gamma)$. *End of Remark 51.*

S.2 Overview of the Analysis

For DeltaNet, we show show BIBS instability (Theorem 57), BIBSD instability (Theorem 58), and BIBLG instability (Theorem 59 and Theorem 60).

The theoretical results are validated experimentally, using the reference implementation of DeltaNet [74] to instantiate the unstable instances identified by the theoretical analyses. In particular, we report figures showing divergent trajectories for states, state derivatives, and loss gradients.

S.3 BIBS Instability of DeltaNet

We show that DeltaNet dynamics do not satisfy wBIBS. In particular, we show that already DeltaNet Units admit divergent state trajectories under mild conditions. The divergent state sequence of an unstable Delta Unit is shown in Figure 4.

Theorem 57 (DeltaNet does not satisfy wBIBS)**.** *Let $\beta \in (0,1)$, and let us define*

$$\tilde{U} := \left\{(1,\beta,k,v) \;\middle|\; k \in K,\, v \in \{0,1\}\right\} \quad \textit{for} \;\; K := \left\{ \frac{k}{\|k\|} \in \mathbb{R}^2 \;\middle|\; k = \begin{pmatrix} 1 \\ \alpha \end{pmatrix},\, \alpha \in [0, 1/2] \right\}.$$

Let $D = \langle X, U, f \mid A, B\rangle$ be a Generic DeltaNet Recurrent Unit with update hyperparameter $c = 1$ and key-value dimensions $d_{\mathrm{kv}} = (2,1)$.

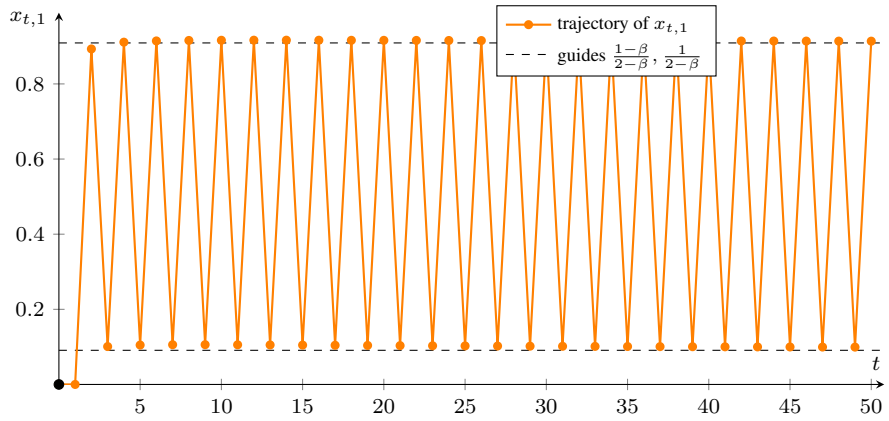
If the state and input spaces satisfy

$$X \neq \emptyset \quad \textit{and} \quad \tilde{U} \subseteq U,$$

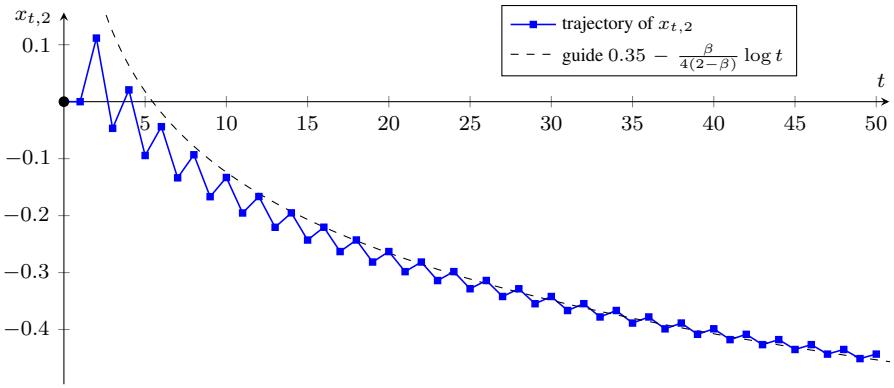
then the DeltaNet Unit D does not satisfy wBIBS. End of Theorem 57.

Proof. Deferred to §S.7.1. □

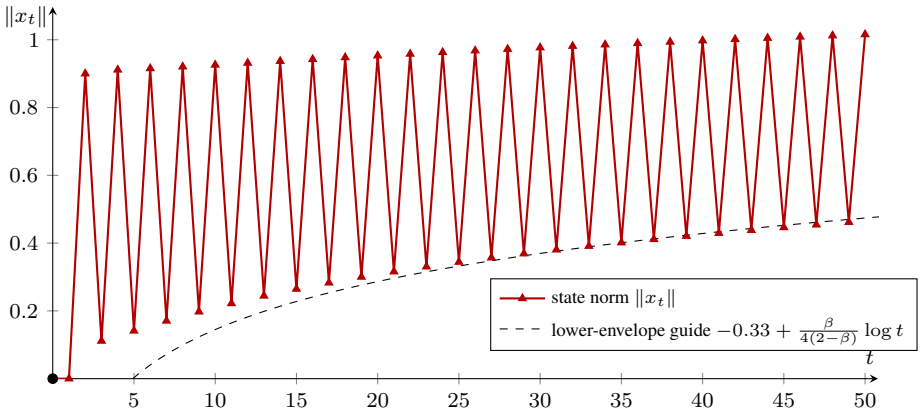
Figure 4: Trajectory of state $x_t = (x_{t,1}, x_{t,2})$ of the divergent DeltaNet dynamics of Lemma 41, computed with the official DeltaNet recurrence (flash-linear-attention), with $\beta = 0.9$ and $x_0 = (0, 0)$. Components are shown in (a) and (b), the state norm in (c).



(a) Component $x_{t,1}$ remains bounded and approaches an alternating two-cycle near $\frac{1-\beta}{2-\beta} \approx 0.0909$ and $\frac{1}{2-\beta} \approx 0.9091$.



(b) Component $x_{t,2}$ drifts downward, consistently with $x_{2n,2} = -\frac{\beta}{4(2-\beta)} \log n + O(1)$.



(c) The state norm $\|x_t\|$ diverges logarithmically; it oscillates with $x_{t,1}$, and its lower envelope (odd t) follows $\frac{\beta}{4(2-\beta)} \log t + O(1)$.

S.4 BIBSD Instability of DeltaNet

Theorem 58 (DeltaNet does not satisfy wBIBSD)**.** *Let $c \in \{1, 2\}$, $\epsilon \in \mathbb{R}^{>0}$, and $\beta_\star \in (0, 1)$.*

Let

$$D_\star = \langle X, U, f; A, B \mid \Lambda_\star \rangle$$

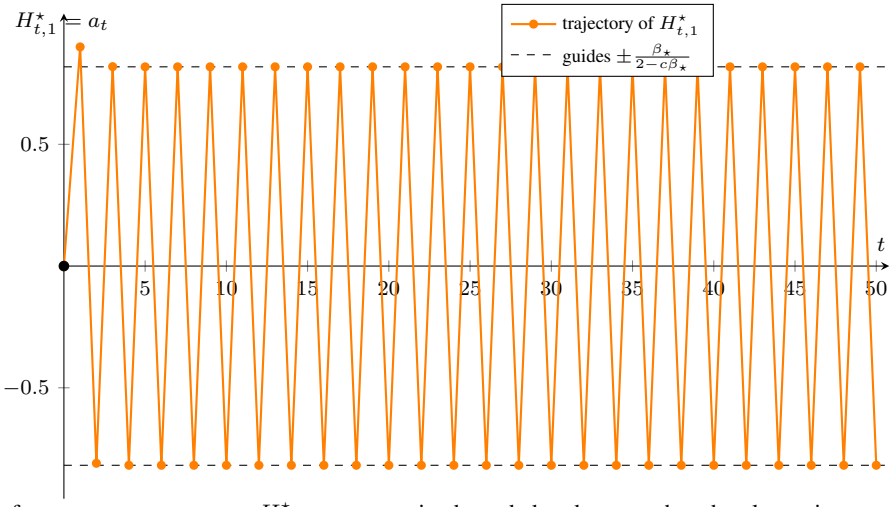
be the specialised parametric DeltaNet Recurrent Unit with hyperparameters $(c, \epsilon, \beta_\star)$ of Construction 1.

Then the DeltaNet Unit $D_\star$ does not satisfy wBIBSD, and consequently it does not satisfy uBIBSD.

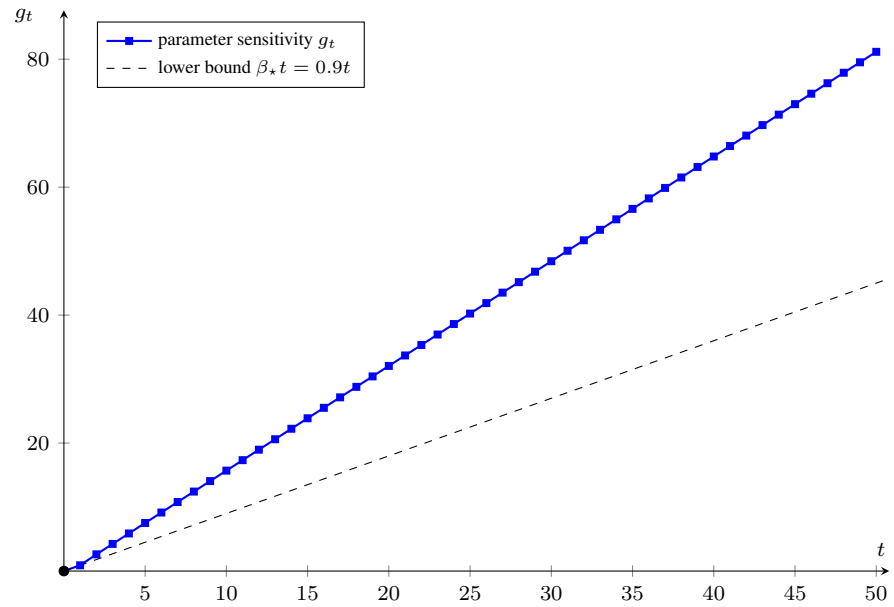
End of Theorem 58.

Proof. Deferred to §S.7.2. □

Figure 5: Bounded state and growing parameter sensitivity for the DeltaNet dynamics of Lemma 42, computed with the official DeltaNet recurrence (flash-linear-attention), with $c = 1$, $\beta_\star = 0.9$, $u_t = (-1)^{t+1}$, $\lambda_\star = 0$, and $x_0 = (0, 0)$. The reference-state trajectory is shown in (a), and its sensitivity with respect to the scalar parameter coordinate λ is shown in (b).

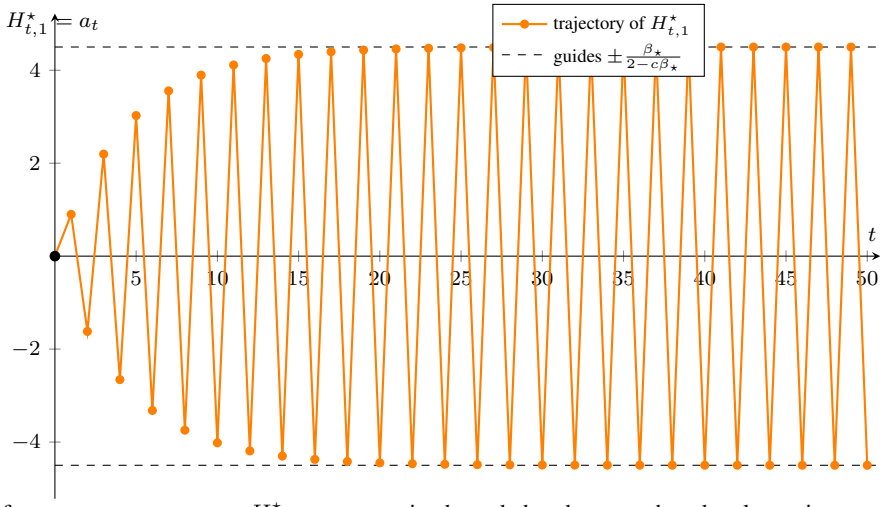


(a) The reference-state component $H^\star_{t,1} = a_t$ remains bounded and approaches the alternating two-cycle $\pm\frac{\beta_\star}{2-c\beta_\star} \approx \pm 0.8182$. The second state component is identically zero.

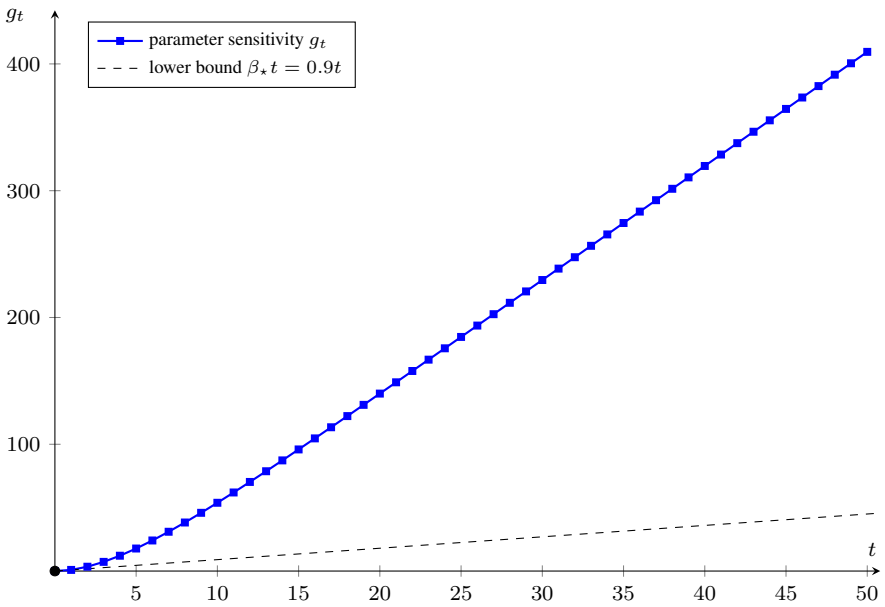


(b) The parameter sensitivity $g_t := \left\| \mathrm{D}_{\lambda_\star} H_t(\lambda_\star) \right\|$ grows without bound and satisfies $g_t \geq \beta_\star t$.

Figure 6: Bounded state and growing parameter sensitivity for the DeltaNet dynamics of Lemma 42, computed with the official DeltaNet recurrence (flash-linear-attention), with $c = 2$, $\beta_\star = 0.9$, $u_t = (-1)^{t+1}$, $\lambda_\star = 0$, and $x_0 = (0, 0)$. The reference-state trajectory is shown in (a), and its sensitivity with respect to the scalar parameter coordinate λ is shown in (b).



(a) The reference-state component $H^\star_{t,1} = a_t$ remains bounded and approaches the alternating two-cycle $\pm\frac{\beta_\star}{2-c\beta_\star} \approx \pm 4.5000$. The second state component is identically zero.



(b) The parameter sensitivity $g_t := \left\|\mathrm{D}_{\lambda_\star} H_t(\lambda_\star)\right\|$ grows without bound and satisfies $g_t \geq \beta_\star t$.

S.5 BIBLG Instability of DeltaNet

We show that DeltaNet's loss gradients are not bounded in general. In particular, we show that already DeltaNet Models consisting of a single recurrent unit show such behaviour, and do so already when their input space consists of two elements only. The proof makes use of state trajectories built in Lemma 42 of §S.6, which are bounded but yield a parameter-sensitivity that grows at least linearly with sequence length.

The formulation of the results makes use of notions introduced in §F.4.

Definition 276 (Nonzero local derivative condition for loss functions)**.** A loss function $\ell : Y \times \hat{Y} \to \mathbb{R}$ with $Y = \mathbb{R}^d$ satisfies the *nonzero local derivative condition* if there exist $a, y_\star \in Y$ and $\hat{y}_\star \in \hat{Y}$ such that

$$a \cdot \mathsf{D}_{y_\star} \ell(y_\star, \hat{y}_\star) \neq 0 \,.$$

In this case, we say that the nonzero local derivative condition is *witnessed by* $(a, y_\star, \hat{y}_\star)$.

End of Definition 276.

Theorem 59 (Linear loss-gradient growth of DeltaNet (already on finite input spaces))**.** *Let $c \in \{1, 2\}$, let $\beta_\star \in (0, 1)$, and let $\ell : Y \times \hat{Y} \to \mathbb{R}$ be an analytic loss function with $Y := \mathbb{R}^d$ and $\hat{Y} \subseteq \mathbb{R}^{d_{\text{target}}}$.*

Suppose that ℓ satisfies the nonzero local derivative condition as witnessed by $(a, y_\star, \hat{y}_\star)$.

For some $\epsilon \in \mathbb{R}^{>0}$, a DeltaNet model $S \in \mathcal{M}[\mathrm{DeltaNet}|c]$, with input space U, output space $Y = \mathbb{R}^d$, parameter space Θ satisfying

$$U_\star := \{-1, 1\} \subseteq U \,, \qquad \Theta_\star \subseteq \Theta \,,$$

where $\Theta_\star$ is as per Construction 1 under the present $\beta_\star$ and ϵ, for which there exist a radius $r \in \mathbb{R}^{>0}$, an input sequence $u \in U_\star^\omega$, a parameter $\theta_\star \in \Theta_\star$, and a constant $C \in \mathbb{R}^{>0}$ such that, for every $T \in \mathbb{N}^{>0}$, we have that

$$z_T := (u_{[1..T]}, \hat{y}_\star)$$

is an admissible sample for S, and additionally:

(1) the sample z_T and parameter $\theta_\star$ belong to their corresponding truncated sets:

$$z_T \in \mathcal{Z}_{\leq r}^{S, \leq T} \,, \qquad \theta_\star \in \Theta_{\leq r} \,;$$

(2) the objective function $\mathcal{L}_{S, z_T}^\ell$ is differentiable at $\theta_\star$;

(3) the loss gradient is bounded from below as follows:

$$\left\| \mathsf{D} \mathcal{L}_{S, z_T}^\ell (\theta_\star) \right\| \geq C \cdot T \,. \tag{193}$$

Consequently, for r the above radius, the growth of the corresponding loss-gradients norm is asymptotically lower-bounded as

$$\mathsf{G}_{\nabla \mathcal{L}}^{\mathrm{S}, \ell, \mathrm{r}}(T) = \Omega(T) \,.$$

In particular, S does not satisfy BIBLG, and hence $\mathcal{M}[\mathrm{DeltaNet}|c]$ does not satisfy BIBLG, even when the input space is restricted to a fixed two-symbol alphabet. End of Theorem 59.

Proof. Deferred to §S.7.3. □

S.5.1 DeltaNet: Divergent Loss-Gradient for Standard Loss Functions

Definition 277 (Standard loss functions)**.** We introduce the following loss functions.

SubDefinition 277.1 (Squared error). The *squared-error loss* is the function

$$\ell_{\text{SE}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{\geq 0} \,, \qquad \ell_{\text{SE}}(y, \hat{y}) := \frac{1}{2} \|y - \hat{y}\|^2 \,.$$

SubDefinition 277.2 (Binary cross-entropy). The *binary cross-entropy loss with logits*, or simply *binary cross-entropy loss*, is the function[20]

$$\ell_{\text{BCE}} : \mathbb{R} \times \{0, 1\} \to \mathbb{R}^{\geq 0} \,, \qquad \ell_{\text{BCE}}(y, \hat{y}) := \log\big(1 + \exp(y)\big) - \hat{y} \cdot y \,.$$

[20] Note that $\sigma(x) := 1/\big(1 + \exp(-x)\big)$ is the logistic sigmoid function.

SubDefinition 277.3 (Cross-entropy). For $K \in \mathbb{N}^{>0}$ with $K \geq 2$, the *K-class softmax cross-entropy loss with logits*, or simply *K-class cross-entropy loss*, is the function

$$\ell_{\text{CE-}K} : \mathbb{R}^K \times [1..K] \to \mathbb{R}^{\geq 0}, \qquad \ell_{\text{CE-}K}(y, k) := -\log\left(\frac{\exp(y_k)}{\sum_{j=1}^{K} \exp(y_j)}\right).$$

We call each of the above loss functions a *standard loss function*. *End of Definition 277.*

Lemma 39 (Standard losses satisfy the nonzero local derivative condition)**.** *Each of the standard loss functions is analytic and hence admissible for every system having matching output space. Furthermore, each of the standard loss functions satisfies the nonzero local derivative condition:*[21]

(1) the squared-error loss $\ell_{\text{SE}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ satisfies the nonzero local derivative condition as witnessed by $(a_{\text{SE}}, y_{\text{SE}}, \hat{y}_{\text{SE}}) := (e_1, e_1, 0)$, and specifically

$$\mathsf{D}_{y_{\text{SE}}} \ell_{\text{SE}}(y_{\text{SE}}, \hat{y}_{\text{SE}}) \cdot a_{\text{SE}} = 1; \tag{194}$$

(2) the binary cross-entropy loss $\ell_{\text{BCE}} : \mathbb{R} \times \{0, 1\} \to \mathbb{R}$ satisfies the nonzero local derivative condition as witnessed by $(a_{\text{BCE}}, y_{\text{BCE}}, \hat{y}_{\text{BCE}}) := (1, 0, 0)$, and specifically

$$\mathsf{D}_{y_{\text{BCE}}} \ell_{\text{BCE}}(y_{\text{BCE}}, \hat{y}_{\text{BCE}}) \cdot a_{\text{BCE}} = \frac{1}{2}; \tag{195}$$

(3) for every $K \in \mathbb{N}^{>0}$ with $K \geq 2$, the K-class cross-entropy loss $\ell_{\text{CE-}K} : \mathbb{R}^K \times [1..K] \to \mathbb{R}$ satisfies the nonzero local derivative condition as witnessed by $(a_{\text{CE}}, y_{\text{CE}}, \hat{y}_{\text{CE}}) := (e_2, 0, 1)$, and specifically

$$\mathsf{D}_{y_{\text{CE}}} \ell_{\text{CE-}K}(y_{\text{CE}}, \hat{y}_{\text{CE}}) \cdot a_{\text{CE}} = \frac{1}{K}. \tag{196}$$

End of Lemma 39.

Proof. Deferred to §S.7.3. □

Theorem 60 (Finite-alphabet loss-gradient growth for standard loss functions)**.** *Let $c \in \{1, 2\}$, let $\beta_\star \in (0, 1)$, and let $U_\star := \{-1, 1\}$. For each of the standard loss functions of Definition 277, there exists a DeltaNet model $S \in \mathcal{M}[\text{DeltaNet}|c]$ having input space $U_\star \subseteq U$ and admitting*

$$\textit{radius } r \in \mathbb{R}^{>0}, \qquad \textit{parameters } \theta_\star \in \Theta_{\leq r}, \qquad \textit{sample } z_T \in \mathcal{Z}_{\leq r}^{S,\leq T} \textit{ for each } T \in \mathbb{T},$$

such that:

$$\left\|\mathsf{D}\mathcal{L}_{S,z_T}^{\ell_{\text{SE}}}(\theta_\star)\right\| \geq \beta_\star \cdot T \qquad \textit{for squared error}, \tag{197}$$

$$\left\|\mathsf{D}\mathcal{L}_{S,z_T}^{\ell_{\text{BCE}}}(\theta_\star)\right\| \geq \frac{\beta_\star}{2} \cdot T \qquad \textit{for binary cross-entropy}, \tag{198}$$

$$\left\|\mathsf{D}\mathcal{L}_{S,z_T}^{\ell_{\text{CE-}K}}(\theta_\star)\right\| \geq \frac{\beta_\star}{K} \cdot T \qquad \textit{for } K\textit{-class cross-entropy, } \forall K \geq 2. \tag{199}$$

Consequently,

- *the loss-gradient growth is*

$$\mathsf{G}_{\nabla\mathcal{L}}^{\text{S},\ell,\text{r}}(T) = \Omega(T) \qquad \forall \ell \in \{\ell_{\text{SE}}, \ell_{\text{BCE}}\} \cup \{\ell_{\text{CE-}K}\}_{K \geq 2},$$

- *the class $\mathcal{M}[\text{DeltaNet}|c]$ does not satisfy BIBLG.*

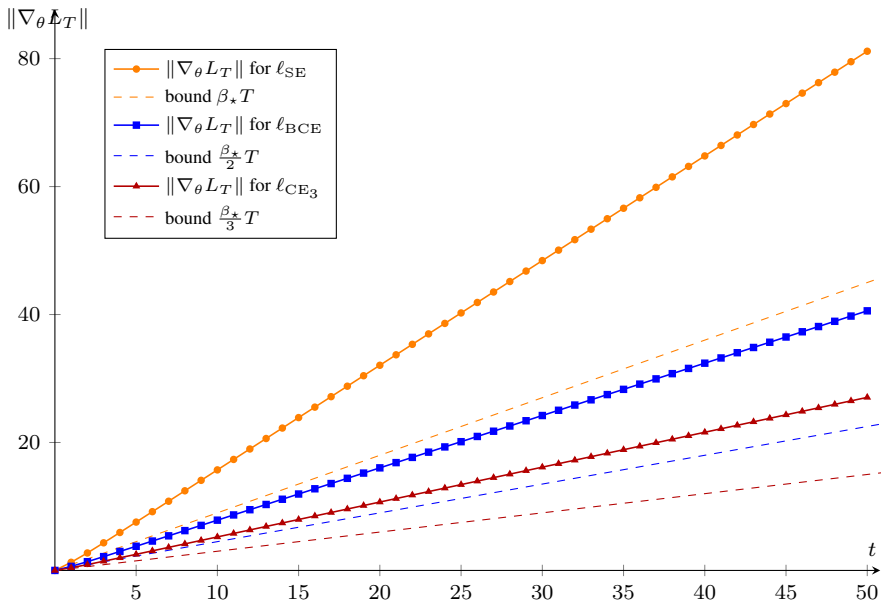
End of Theorem 60.

Proof. Deferred to §S.7.3. □

Remark 52. Theorem 60 shows that DeltaNet models can exhibit an unbounded loss gradient even when their input space consists of two elements, for any choice a loss function among the standard ones we have considered. *End of Remark 52.*

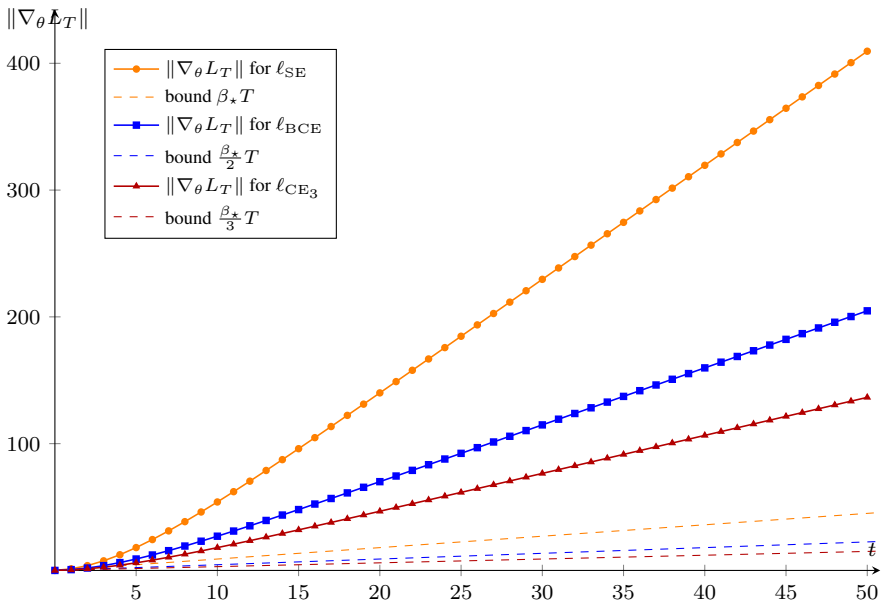
[21] The *nonzero local derivative condition* is introduced in Definition 276.

Figure 7: Loss-gradient norms $\|\nabla_\theta L_T(\theta_\star)\|$ of the one-unit DeltaNet model of Theorem 59 on input prefixes $u_{1:T}$, $u_t = (-1)^{t+1}$, for the squared-error, binary cross-entropy and 3-class cross-entropy losses (Theorem 60), computed by automatic differentiation through the official DeltaNet recurrence (flash-linear-attention), with $c = 1$ and $\beta_\star = 0.9$. Dashed lines are the lower bounds $\beta_\star T$, $\frac{\beta_\star}{2} T$, and $\frac{\beta_\star}{3} T$, respectively.



(a) The loss-gradient norm grows at least linearly in the sequence length T, for every standard loss.

Figure 8: Loss-gradient norms $\|\nabla_\theta L_T(\theta_\star)\|$ of the one-unit DeltaNet model of Theorem 59 on input prefixes $u_{1:T}$, $u_t = (-1)^{t+1}$, for the squared-error, binary cross-entropy and 3-class cross-entropy losses (Theorem 60), computed by automatic differentiation through the official DeltaNet recurrence (flash-linear-attention), with $c = 2$ and $\beta_\star = 0.9$. Dashed lines are the lower bounds $\beta_\star T$, $\frac{\beta_\star}{2} T$, and $\frac{\beta_\star}{3} T$, respectively.



(a) The loss-gradient norm grows at least linearly in the sequence length T, for every standard loss.

S.6 DeltaNet: Technical Lemmas on Instability

S.6.1 DeltaNet: Divergent state trajectories

Lemma 40 (*Aux. for Lemma 41:* Initial-state comparison for nonexpansive affine recurrences)**.** *Let us consider a sequence of vectors $(b_t)_{t\in\mathbb{T}}$ and a sequence of matrices $(A_t)_{t\in\mathbb{T}}$ such that, for every $t \in \mathbb{T}$,*

$$b_t \in \mathbb{R}^d, \qquad A_t \in \mathbb{R}^{d\times d} \quad \textit{with} \quad \|A_t\| \le 1.$$

Let $x_0, \tilde{x}_0 \in \mathbb{R}^d$, and let the sequences $(x_t)_{t\in\mathbb{N}}$ and $(\tilde{x}_t)_{t\in\mathbb{N}}$ be defined as follows, for every $t \in \mathbb{T}$,

$$x_t \;:=\; A_t \cdot x_{t-1} + b_t, \qquad \tilde{x}_t \;:=\; A_t \cdot \tilde{x}_{t-1} + b_t.$$

Then, for every $t \in \mathbb{N}$,

$$\|x_t - \tilde{x}_t\| \;\le\; \|x_0 - \tilde{x}_0\|. \tag{200}$$

Consequently, if $\|\tilde{x}_t\| \to \infty$ as $t \to \infty$, then $\|x_t\| \to \infty$ as $t \to \infty$. End of Lemma 40.

Lemma 41 (*Aux. for Thm. 57:* Divergent state trajectories of DeltaNet)**.** *Let $\beta \in (0,1)$, and let us define the following quantities for all $t \ge 1$,*

$$\beta_t := \beta \qquad \epsilon_t := \begin{cases} \frac{1}{t+2} & t \textit{ odd}, \\ \frac{1}{2(t+2)} & t \textit{ even}, \end{cases} \qquad k_t := \frac{1}{\sqrt{1+\epsilon_t^2}} \cdot \begin{pmatrix} 1 \\ \epsilon_t \end{pmatrix} \qquad v_t := \begin{cases} 0, & t \textit{ odd}, \\ 1, & t \textit{ even}. \end{cases}$$

Let $x_0 := (0,0)$ and let $(x_t)_{t\ge1}$ be the state sequence defined by the recurrence

$$x_t \;:=\; \left(I - \beta_t \cdot k_t \cdot k_t^\top\right) \cdot x_{t-1} \;+\; \beta_t \cdot k_t \cdot v_t^\top \,.$$

Then $\|x_t\| \to \infty$ as $t \to \infty$. In particular, the values $x_{2n} = (x_{2n,1}, x_{2n,2})$ at even steps and the values $x_{2n-1} = (x_{2n-1,1}, x_{2n-1,2})$ at odd steps are as follows,

$$x_{2n,1} = \frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right), \qquad x_{2n-1,1} = \frac{1-\beta}{2-\beta} + O\left(\frac{\log n}{n}\right),$$

$$x_{2n,2} = -\frac{\beta}{4(2-\beta)}\log n + O(1), \qquad x_{2n-1,2} = -\frac{\beta}{4(2-\beta)}\log n + O(1).$$

Note that $\frac{\beta}{4(2-\beta)} > 0$. End of Lemma 41.

S.6.2 DeltaNet: Divergent parameter-sensitivity of DeltaNet on finite input spaces

Construction 1 (*Aux. for Lemma 42:* Specialised DeltaNet parametrisation)**.** Let $c \in \{1,2\}$ $\epsilon \in \mathbb{R}^{>0}$, and $\beta_\star \in (0,1)$.

We define the *specialised parametric DeltaNet Recurrent Unit* with hyperparameters $(c, \epsilon, \beta_\star)$ as the parametric Left-Multiplicative Linear Recurrent Unit

$$D_\star := \langle X, U, f; A, B \,|\, \Lambda_\star \rangle \qquad \text{with} \qquad d_{\text{in}} = 1\,, \quad d_{\text{k}} = 2\,, \quad d_{\text{v}} = 1\,,$$

and components as follows: as follows:

- state, input, and parameter spaces given respectively by
$$X := \mathbb{R}^{2\times1} \cong \mathbb{R}^2\,, \qquad U := \{-1, 1\}\,, \qquad \Lambda_\star \;:=\; \mathrm{Ball}_{\mathbb{R}}(\lambda_\star, \epsilon) \quad \text{with} \quad \lambda_\star \;=\; 0\,;$$
- the parameter values induced by $\lambda \in \Lambda_\star$ are
$$W_{\text{k}}[\lambda] := \begin{pmatrix} 0 \\ \lambda \end{pmatrix}, \quad W_{\text{v}} := 1, \quad w_\beta := 0, \quad b_{\text{k}} := e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad b_{\text{v}} := 0, \quad b_\beta := \log\left(\frac{\beta_\star}{1-\beta_\star}\right);$$
- the recurrence-input functions of are

$\tilde{k} : U \times \Lambda_\star \to \mathbb{R}^2$	$\tilde{k}(u;\lambda) := W_{\text{k}}[\lambda] \cdot u + b_{\text{k}}$	*key preactivation*,
$k : U \times \Lambda_\star \to \mathbb{S}^1$	$k(u;\lambda) := \tilde{k}(u;\lambda) / \big\|\tilde{k}(u;\lambda)\big\|$	*key*,
$\beta : U \times \Lambda_\star \to (0,1)$	$\beta(u;\lambda) := \sigma\big(w_\beta \cdot u + b_\beta\big)$	*write strength*,
$v : U \times \Lambda_\star \to \mathbb{R}$	$v(u;\lambda) := W_{\text{v}} \cdot u + b_{\text{v}}$	*value*;

- the state-update matrix and additive term are

$$A(u;\lambda) := I_2 - c\cdot\beta(u;\lambda)\cdot k(u;\lambda)\cdot k(u;\lambda)^\top \quad \text{and} \quad B(u;\lambda) \quad := \beta(u;\lambda)\cdot k(u;\lambda)\cdot v(u;\lambda)\,.$$

Thus, the dynamics function is

$$f(H,u;\lambda) \;=\; A(u;\lambda)\cdot H + B(u;\lambda)\,.$$

Fact 60 (The specialised parametrisation is DeltaNet)**.** *The* specialised parametric DeltaNet Recurrent Unit $D_\star$ *with hyperparameters* $(c,\epsilon,\beta_\star)$ *of Construction 1 is a DeltaNet Recurrent Unit with no decay gate and update hyperparameter* c*, under the specialised parametrisation*

$$\vartheta \;:\; \lambda \longmapsto \Big(\big(W_{\mathrm{k}}[\lambda], b_{\mathrm{k}}\big),\ \big(W_{\mathrm{v}}, b_{\mathrm{v}}\big),\ \big(w_\beta, b_\beta\big)\Big) \qquad \textit{for} \quad \lambda\in\Lambda_\star\,.$$

Specifically, the standard DeltaNet Recurrent Unit with no decay gate, update hyperparameter c*, and parameter space*

$$\Theta_\star \;:=\; \mathrm{Im}(\vartheta) \;=\; \big\{\theta[\lambda] : \lambda\in\Lambda_\star\big\}$$

has precisely the dynamics of $D_\star$ *under the parameter identification* ϑ*.* End of Fact 60.

Remark 53. Under the specialised parametrisation of Fact 60, the scalar parameter λ is precisely the $(2,1)$-entry of $W_{\mathrm{k}}[\lambda]$, while every remaining standard DeltaNet parameter coordinate is constant.
End of Remark 53.

Lemma 42 (*Aux. for Thms. 58, 59:* Parameter-sensitivity growth of DeltaNet on finite input spaces)**.** *Let* $c\in\{1,2\}$*,* $\epsilon\in\mathbb{R}^{>0}$*, and* $\beta_\star\in(0,1)$*, and let*

$$D_\star = \langle X, U, f; A, B \,|\, \Lambda_\star\rangle$$

be the specialised parametric DeltaNet Recurrent Unit with hyperparameters $(c,\epsilon,\beta_\star)$ *as per Construction 1. Let*

$$x_0 := (0,0)\,, \qquad\qquad u := (u_t)_{t\in\mathbb{T}} \in U^\omega \quad \textit{with} \quad u_t := (-1)^{t+1}\,.$$

For every $t\in\mathbb{T}$*, let us define the parameter-to-state function*

$$H_t : \Lambda_\star \to \mathbb{R}^2, \qquad\qquad H_t(\lambda) := \check{D}_\star[x_0,u,t]\big(\lambda\big)\,,$$

and let $H_t^\star := H_t(\lambda_\star)$ *for* $\lambda_\star := 0$*.*

Then the following holds:

- *the state sequence* $(H_t^\star)_{t\in\mathbb{T}}$ *is bounded:*

$$\sup_{t\in\mathbb{N}} \big\|H_t^\star\big\| < \infty\,; \tag{201}$$

- *for every* $t\in\mathbb{T}$*, the parameter-to-state function* H_t *is differentiable at* $\lambda_\star$*, and it satisfies*

$$\big\|\mathsf{D}H_t(\lambda_\star)\big\| \;\geq\; \beta_\star\cdot t\,; \tag{202}$$

- *consequently,*

$$\big\|\mathsf{D}H_t(\lambda_\star)\big\| \to \infty \quad \textit{as} \quad t\to\infty\,.$$

End of Lemma 42.

S.7 DeltaNet: Deferred Proofs

S.7.1 BIBS Instability of DeltaNet

Lemma 43 (*Aux. for Lemma 41:* Initial-state comparison for nonexpansive affine recurrences)**.** *Let us consider a sequence of vectors* $(b_t)_{t\in\mathbb{T}}$ *and a sequence of matrices* $(A_t)_{t\in\mathbb{T}}$ *such that, for every* $t\in\mathbb{T}$*,*

$$b_t\in\mathbb{R}^d, \qquad\qquad A_t\in\mathbb{R}^{d\times d} \quad \textit{with} \quad \|A_t\|\leq 1.$$

Let $x_0,\tilde{x}_0\in\mathbb{R}^d$*, and let the sequences* $(x_t)_{t\in\mathbb{N}}$ *and* $(\tilde{x}_t)_{t\in\mathbb{N}}$ *be defined as follows, for every* $t\in\mathbb{T}$*,*

$$x_t \;:=\; A_t\cdot x_{t-1} + b_t, \qquad\qquad \tilde{x}_t \;:=\; A_t\cdot\tilde{x}_{t-1} + b_t.$$

Then, for every $t\in\mathbb{N}$*,*

$$\|x_t - \tilde{x}_t\| \;\leq\; \|x_0 - \tilde{x}_0\|. \tag{200}$$

Consequently, if $\|\tilde{x}_t\|\to\infty$ *as* $t\to\infty$*, then* $\|x_t\|\to\infty$ *as* $t\to\infty$*.* End of Lemma 43.

Proof. We prove Equation (200) by induction on $t \in \mathbb{N}$.

For $t = 0$, the claim holds trivially, since

$$\|x_0 - \tilde{x}_0\| \le \|x_0 - \tilde{x}_0\|.$$

Now let $t \in \mathbb{T}$, and suppose that

$$\|x_{t-1} - \tilde{x}_{t-1}\| \le \|x_0 - \tilde{x}_0\|.$$

By the definitions of x_t and $\tilde{x}_t$,

$$\begin{aligned} x_t - \tilde{x}_t &= \big(A_t \cdot x_{t-1} + b_t\big) - \big(A_t \cdot \tilde{x}_{t-1} + b_t\big) \\ &= A_t \cdot \big(x_{t-1} - \tilde{x}_{t-1}\big). \end{aligned}$$

Therefore, by the defining property of the induced matrix norm, the assumption $\|A_t\| \le 1$, and the induction hypothesis,

$$\begin{aligned} \|x_t - \tilde{x}_t\| &\le \|A_t\| \cdot \|x_{t-1} - \tilde{x}_{t-1}\| \\ &\le \|x_{t-1} - \tilde{x}_{t-1}\| \\ &\le \|x_0 - \tilde{x}_0\|. \end{aligned}$$

This proves Equation (200) for every $t \in \mathbb{N}$.

For the consequent claim, the reverse triangle inequality and Equation (200) give

$$\begin{aligned} \|x_t\| &\ge \|\tilde{x}_t\| - \|x_t - \tilde{x}_t\| \\ &\ge \|\tilde{x}_t\| - \|x_0 - \tilde{x}_0\| \end{aligned}$$

for every $t \in \mathbb{N}$. Since $\|x_0 - \tilde{x}_0\|$ is independent of t, it follows that

$$\|\tilde{x}_t\| \to \infty \quad \Longrightarrow \quad \|x_t\| \to \infty \qquad \text{as } t \to \infty.$$

□

Lemma 44 (Discrete Gronwall estimate)**.** *Let $(a_t)_{t \ge 0}$ be a nonnegative sequence satisfying*

$$a_t \le u_t a_{t-1} + b_t, \qquad t \ge 1,$$

where $u_t \ge 0$ and $b_t \ge 0$. Then

$$a_t \le a_0 \prod_{j=1}^{t} u_j + \sum_{i=1}^{t} b_i \prod_{j=i+1}^{t} u_j.$$

In particular, if

$$u_t = 1 + \eta_t, \qquad \eta_t \ge 0,$$

then

$$a_t \le \left(a_0 + \sum_{i=1}^{t} b_i\right) \exp\left(\sum_{j=1}^{t} \eta_j\right).$$

Proof. Iterating the recursion gives

$$a_t \le a_0 \prod_{j=1}^{t} u_j + \sum_{i=1}^{t} b_i \prod_{j=i+1}^{t} u_j.$$

This proves the first claim. If $u_j = 1 + \eta_j$, then

$$\prod_{j=i+1}^{t} u_j = \prod_{j=i+1}^{t} (1 + \eta_j) \le \exp\left(\sum_{j=i+1}^{t} \eta_j\right) \le \exp\left(\sum_{j=1}^{t} \eta_j\right),$$

using $1 + x \le e^x$ for $x \ge 0$. Therefore

$$a_t \le \left(a_0 + \sum_{i=1}^{t} b_i\right) \exp\left(\sum_{j=1}^{t} \eta_j\right).$$

□

Lemma 45 (*Aux. for Thm. 57:* Divergent state trajectories of DeltaNet)**.** *Let $\beta \in (0,1)$, and let us define the following quantities for all $t \geq 1$,*

$$\beta_t := \beta \qquad \epsilon_t := \begin{cases} \frac{1}{t+2} & t \text{ odd}, \\ \frac{1}{2(t+2)} & t \text{ even}, \end{cases} \qquad k_t := \frac{1}{\sqrt{1+\epsilon_t^2}} \cdot \begin{pmatrix} 1 \\ \epsilon_t \end{pmatrix} \qquad v_t := \begin{cases} 0, & t \text{ odd}, \\ 1, & t \text{ even}. \end{cases}$$

Let $x_0 := (0,0)$ and let $(x_t)_{t\geq 1}$ be the state sequence defined by the recurrence

$$x_t \;:=\; \left(I - \beta_t \cdot k_t \cdot k_t^\top\right) \cdot x_{t-1} \;+\; \beta_t \cdot k_t \cdot v_t^\top \;.$$

Then $\|x_t\| \to \infty$ as $t \to \infty$. In particular, the values $x_{2n} = (x_{2n,1}, x_{2n,2})$ at even steps and the values $x_{2n-1} = (x_{2n-1,1}, x_{2n-1,2})$ at odd steps are as follows,

$$x_{2n,1} = \frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right), \qquad x_{2n-1,1} = \frac{1-\beta}{2-\beta} + O\left(\frac{\log n}{n}\right),$$

$$x_{2n,2} = -\frac{\beta}{4(2-\beta)} \log n + O(1), \qquad x_{2n-1,2} = -\frac{\beta}{4(2-\beta)} \log n + O(1).$$

Note that $\frac{\beta}{4(2-\beta)} > 0$. End of Lemma 45.

Proof. The recurrence can be equivalently written as

$$x_t = x_{t-1} + \beta \cdot k_t \cdot \left(v_t - k_t^\top \cdot x_{t-1}\right) .$$

Also, using the quantities

$$x_t := \begin{pmatrix} x_{t,1} \\ x_{t,2} \end{pmatrix}, \qquad k_t := \begin{pmatrix} c_t \\ s_t \end{pmatrix}, \qquad c_t := \frac{1}{\sqrt{1+\epsilon_t^2}}, \qquad s_t := \frac{\epsilon_t}{\sqrt{1+\epsilon_t^2}},$$

it can be written as

$$x_t = x_{t-1} + \beta \cdot \begin{pmatrix} c_t \\ s_t \end{pmatrix} \cdot \left(v_t \;-\; c_t \cdot x_{t-1,1} \;-\; s_t \cdot x_{t-1,2}\right),$$

and component-wise as

$$x_{t,1} = x_{t-1,1} + \beta \cdot c_t \cdot \left(v_t \;-\; c_t \cdot x_{t-1,1} \;-\; s_t \cdot x_{t-1,2}\right),$$

$$x_{t,2} = x_{t-1,2} + \beta \cdot s_t \cdot \left(v_t \;-\; c_t \cdot x_{t-1,1} \;-\; s_t \cdot x_{t-1,2}\right).$$

Bound on overall state. We first derive an upper bound on the state. Since

$$\left\|I - \beta \cdot k_t k_t^\top\right\| \leq 1, \qquad \|k_t\| = 1,$$

we have

$$\|x_t\| \leq \|x_{t-1}\| + \beta \cdot |v_t| \leq \|x_{t-1}\| + \beta.$$

Since $x_0 = (0,0)$, this gives

$$\|x_t\| \leq \beta \cdot t,$$

and in turn

$$|x_{t,1}| \leq \beta \cdot t, \qquad |x_{t,2}| \leq \beta \cdot t.$$

Bound on first state component. We now derive a uniform upper bound on the first state component. We expand the recurrence for the first state component:

$$x_{t,1} = \left(1 - \beta \cdot c_t^2\right) \cdot x_{t-1,1} - \beta \cdot c_t \cdot s_t \cdot x_{t-1,2} + \beta \cdot c_t \cdot v_t.$$

Taking absolute values and using $0 < c_t \leq 1$, $|v_t| \leq 1$, and $s_t \leq \epsilon_t \leq \frac{1}{t+2}$, we obtain

$$|x_{t,1}| \leq \left(1 - \beta \cdot c_t^2\right) \cdot |x_{t-1,1}| + \beta \cdot s_t \cdot |x_{t-1,2}| + \beta.$$

Since $\epsilon_t \le \frac{1}{t+2} \le \frac{1}{3}$,

$$c_t^2 = \frac{1}{1+\epsilon_t^2} \ge \frac{1}{1+\frac{1}{9}} = \frac{9}{10}.$$

Thus

$$s_t \cdot |x_{t-1,2}| \le \frac{1}{t+2} \cdot \beta \cdot (t-1) \le \beta,$$

and hence

$$|x_{t,1}| \le \left(1 - \frac{9}{10} \cdot \beta\right) \cdot |x_{t-1,1}| + \beta^2 + \beta.$$

Iterating this scalar inequality yields

$$\begin{aligned} |x_{t,1}| &\le (\beta + \beta^2) \cdot \sum_{j=0}^{t-1} \left(1 - \frac{9}{10} \cdot \beta\right)^j \le (\beta + \beta^2) \cdot \frac{1}{\frac{9}{10} \cdot \beta} \\ &= \frac{10}{9} \cdot (1+\beta) =: M. \end{aligned}$$

Namely, M is a bound on $|x_{t,1}|$ holding uniformly for all time steps t.

Asymptotic upper bound on second state component. We now show an asymptotic upper bound on the second state component. We expand the recurrence for the second state component as follows:

$$x_{t,2} = x_{t-1,2} + \beta \cdot s_t \cdot (v_t - c_t \cdot x_{t-1,1}) - \beta \cdot s_t^2 \cdot x_{t-1,2}.$$

Taking absolute values gives

$$|x_{t,2}| \le |x_{t-1,2}| + \beta \cdot s_t \cdot |v_t - c_t \cdot x_{t-1,1}| + \beta \cdot s_t^2 \cdot |x_{t-1,2}|.$$

Since $|v_t| \le 1$ and $0 < c_t \le 1$,

$$|v_t - c_t \cdot x_{t-1,1}| \le |v_t| + c_t \cdot |x_{t-1,1}| \le 1 + M.$$

Since

$$0 < s_t = \frac{\epsilon_t}{\sqrt{1+\epsilon_t^2}} \le \epsilon_t,$$

we have $s_t^2 \le \epsilon_t^2$. Thus

$$|x_{t,2}| \le |x_{t-1,2}| + \beta \cdot (1+M) \cdot \epsilon_t + \beta \cdot \epsilon_t^2 \cdot |x_{t-1,2}|,$$

or equivalently

$$|x_{t,2}| \le \left(1 + \beta \cdot \epsilon_t^2\right) \cdot |x_{t-1,2}| + \beta \cdot (1+M) \cdot \epsilon_t.$$

Letting

$$C := \max\left(\beta,\, \beta \cdot (1+M)\right),$$

the above yields

$$|x_{t,2}| \le (1 + C \cdot \epsilon_t^2) \cdot |x_{t-1,2}| + C \cdot \epsilon_t.$$

We apply the discrete Gronwall estimate, Lemma 44, with

$$a_t = |x_{t,2}|, \qquad u_t = 1 + C \cdot \epsilon_t^2, \qquad b_t = C \cdot \epsilon_t.$$

This gives

$$|x_{t,2}| \le \left(|x_{0,2}| + C \cdot \sum_{i=1}^{t} \epsilon_i\right) \cdot \exp\left(C \cdot \sum_{i=1}^{t} \epsilon_i^2\right).$$

We have

$$\sum_{i=1}^{t} \epsilon_i = O\left(\sum_{i=1}^{t} \frac{1}{i}\right) = O(\log t),$$
$$\sum_{i=1}^{\infty} \epsilon_i^2 < \infty.$$

Therefore

$$|x_{t,2}| = O(\log t), \qquad \epsilon_t \cdot x_{t,2} = O\left(\frac{\log t}{t}\right) \to 0.$$

Convergence of first component. We now prove

$$x_{2n,1} = \frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right), \qquad x_{2n-1,1} = \frac{1-\beta}{2-\beta} + O\left(\frac{\log n}{n}\right).$$

Recall that

$$x_{t,1} = \left(1 - \beta \cdot c_t^2\right) \cdot x_{t-1,1} - \beta \cdot c_t \cdot s_t \cdot x_{t-1,2} + \beta \cdot c_t \cdot v_t.$$

Since

$$c_t = 1 + O(\epsilon_t^2), \qquad s_t = O(\epsilon_t), \qquad \epsilon_t = O\left(\frac{1}{t}\right),$$

and since $|x_{t,1}| \le M$ and $|x_{t,2}| = O(\log t)$, there exists a sequence $(\theta_t)_{t\ge1}$ and a constant $C' > 0$ such that

$$x_{t,1} = (1-\beta) \cdot x_{t-1,1} + \beta \cdot v_t + \theta_t, \qquad |\theta_t| \le C' \cdot \frac{\log t}{t}.$$

Since $v_{2n-1} = 0$ and $v_{2n} = 1$, we obtain

$$x_{2n-1,1} = (1-\beta) \cdot x_{2n-2,1} + \theta_{2n-1},$$
$$x_{2n,1} = (1-\beta) \cdot x_{2n-1,1} + \beta + \theta_{2n}.$$

Substituting the first relation into the second gives

$$x_{2n,1} = (1-\beta)^2 \cdot x_{2n-2,1} + \beta + (1-\beta) \cdot \theta_{2n-1} + \theta_{2n}.$$

Letting $y_n := x_{2n,1}$, we have

$$y_n = (1-\beta)^2 \cdot y_{n-1} + \beta + \delta_n, \qquad \delta_n := (1-\beta) \cdot \theta_{2n-1} + \theta_{2n}.$$

Since $|\delta_n| \le C' \cdot \frac{\log n}{n}$, and since the affine map

$$y \mapsto (1-\beta)^2 \cdot y + \beta$$

has fixed point $\frac{1}{2-\beta}$, the same geometric-series estimate as in the proof of a stable affine recursion gives

$$x_{2n,1} = \frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right).$$

Using

$$x_{2n-1,1} = (1-\beta) \cdot x_{2n-2,1} + \theta_{2n-1},$$

we also obtain

$$x_{2n-1,1} = \frac{1-\beta}{2-\beta} + O\left(\frac{\log n}{n}\right).$$

Divergence of second component. We derive the two-step change for the second state component. Since $v_{2n-1} = 0$ and $v_{2n} = 1$, we have

$$\begin{aligned} x_{2n,2} - x_{2n-2,2} = {} & \beta \cdot s_{2n-1} \left(0 - c_{2n-1} \cdot x_{2n-2,1} - s_{2n-1} \cdot x_{2n-2,2}\right) \\ & + \beta \cdot s_{2n} \left(1 - c_{2n} \cdot x_{2n-1,1} - s_{2n} \cdot x_{2n-1,2}\right). \end{aligned}$$

Using

$$c_t = 1 + O\left(\frac{1}{n^2}\right), \qquad s_t = \epsilon_t + O\left(\frac{1}{n^3}\right),$$

for $t \sim n$, together with $x_{t,1} = O(1)$ and $x_{t,2} = O(\log t)$, we obtain

$$x_{2n,2} - x_{2n-2,2} = -\beta \cdot \epsilon_{2n-1} \cdot x_{2n-2,1} + \beta \cdot \epsilon_{2n} \cdot (1 - x_{2n-1,1}) + O\left(\frac{\log n}{n^2}\right).$$

Now substitute

$$x_{2n-2,1} = \frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right), \qquad x_{2n-1,1} = \frac{1-\beta}{2-\beta} + O\left(\frac{\log n}{n}\right).$$

Thus

$$1 - x_{2n-1,1} = \frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right),$$

and hence

$$\begin{aligned} x_{2n,2} - x_{2n-2,2} = &-\beta \cdot \epsilon_{2n-1} \left(\frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right)\right) \\ &+ \beta \cdot \epsilon_{2n} \left(\frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right)\right) + O\left(\frac{\log n}{n^2}\right). \end{aligned}$$

Since

$$\begin{aligned} \epsilon_{2n-1} &= \frac{1}{2n+1} = \frac{1}{2n} + O\left(\frac{1}{n^2}\right), \\ \epsilon_{2n} &= \frac{1}{2(2n+2)} = \frac{1}{4n} + O\left(\frac{1}{n^2}\right), \end{aligned}$$

we get

$$\begin{aligned} x_{2n,2} - x_{2n-2,2} &= -\beta \cdot \frac{1}{2n} \cdot \frac{1}{2-\beta} + \beta \cdot \frac{1}{4n} \cdot \frac{1}{2-\beta} + O\left(\frac{\log n}{n^2}\right) \\ &= -\frac{\beta}{4(2-\beta)} \cdot \frac{1}{n} + O\left(\frac{\log n}{n^2}\right). \end{aligned}$$

Finally,

$$\begin{aligned} x_{2n,2} &= x_{2n,2} - x_{0,2} && \text{| } x_{0,2} = 0 \text{ by assumption} \\ &= \sum_{m=1}^{n} \left(x_{2m,2} - x_{2(m-1),2}\right) && \text{| telescopic sum} \\ &= \sum_{m=1}^{n} \left(-\frac{\beta}{4(2-\beta)} \cdot \frac{1}{m} + O\left(\frac{\log m}{m^2}\right)\right) \\ &= -\frac{\beta}{4(2-\beta)} \cdot \sum_{m=1}^{n} \frac{1}{m} + \sum_{m=1}^{n} O\left(\frac{\log m}{m^2}\right). \end{aligned}$$

Since

$$\sum_{m=1}^{n} \frac{1}{m} = \log n + O(1), \qquad \sum_{m=1}^{\infty} \frac{\log m}{m^2} < \infty,$$

we obtain

$$x_{2n,2} = -\frac{\beta}{4(2-\beta)} \log n + O(1).$$

In particular, $x_{2n,2} \to -\infty$, and hence $\|x_t\| \to \infty$.

Second component at odd steps. From the odd update,

$$x_{2n-1,2} - x_{2n-2,2} = \beta \cdot s_{2n-1} \left(0 - c_{2n-1} \cdot x_{2n-2,1} - s_{2n-1} \cdot x_{2n-2,2}\right).$$

Using the same estimates as above,

$$\begin{aligned} x_{2n-1,2} - x_{2n-2,2} &= -\beta \cdot \epsilon_{2n-1} \cdot x_{2n-2,1} + O\left(\frac{\log n}{n^2}\right) \\ &= -\beta \cdot \left(\frac{1}{2n} + O\left(\frac{1}{n^2}\right)\right)\left(\frac{1}{2-\beta} + O\left(\frac{\log n}{n}\right)\right) + O\left(\frac{\log n}{n^2}\right) \\ &= -\frac{\beta}{2(2-\beta)} \cdot \frac{1}{n} + O\left(\frac{\log n}{n^2}\right). \end{aligned}$$

Therefore

$$x_{2n-1,2} = -\frac{\beta}{4(2-\beta)} \log n + O(1).$$

This concludes the proof. □

Theorem 61 (DeltaNet does not satisfy wBIBS)**.** *Let $\beta \in (0,1)$, and let us define*

$$\tilde{U} := \left\{(1, \beta, k, v) \;\middle|\; k \in K,\, v \in \{0,1\}\right\} \quad \textit{for} \;\; K := \left\{ \frac{k}{\|k\|} \in \mathbb{R}^2 \;\middle|\; k = \begin{pmatrix} 1 \\ \alpha \end{pmatrix},\, \alpha \in [0, 1/2] \right\}.$$

Let $D = \langle X, U, f \mid A, B\rangle$ be a Generic DeltaNet Recurrent Unit with update hyperparameter $c = 1$ and key-value dimensions $d_{\mathrm{kv}} = (2,1)$.

If the state and input spaces satisfy

$$X \neq \emptyset \qquad \textit{and} \qquad \tilde{U} \subseteq U,$$

then the DeltaNet Unit D does not satisfy wBIBS. End of Theorem 61.

Proof. Consider the sequences $(\beta_t)_{t\in\mathbb{T}}$, $(k_t)_{t\in\mathbb{T}}$, and $(v_t)_{t\in\mathbb{T}}$ defined in Lemma 41, and define the input sequence $(u_t)_{t\in\mathbb{T}}$ by

$$u_t := (1, \beta_t, k_t, v_t) = (1, \beta, k_t, v_t) \qquad \forall t \in \mathbb{T}.$$

By the definitions in Lemma 41, we have

$$k_t = \frac{1}{\sqrt{1+\epsilon_t^2}} \cdot \begin{pmatrix} 1 \\ \epsilon_t \end{pmatrix}, \qquad \epsilon_t \in [0, 1/2], \qquad v_t \in \{0,1\},$$

and hence

$$k_t \in K \qquad \text{and} \qquad u_t \in \tilde{U} \subseteq U \qquad \forall t \in \mathbb{T}.$$

Therefore, $(u_t)_{t\in\mathbb{T}}$ is an admissible input sequence for D. Moreover, it is bounded, since

$$g_t = 1, \qquad \beta_t = \beta, \qquad \|k_t\| = 1, \qquad |v_t| \leq 1 \qquad \forall t \in \mathbb{T}.$$

Since $X \neq \emptyset$, fix any

$$x_0 \in X,$$

and let $(x_t)_{t\in\mathbb{T}}$ be the corresponding state sequence under $(u_t)_{t\in\mathbb{T}}$,

$$x_t := f(x_{t-1}, u_t) \qquad \forall t \in \mathbb{T}.$$

Since D has update hyperparameter $c = 1$ and the decay-gate component of every u_t is $g_t = 1$, Definition 274 gives

$$x_t \;=\; A_t \cdot x_{t-1} + b_t, \tag{203}$$

where

$$A_t \;:=\; I_2 - \beta \cdot k_t \cdot k_t^\top, \qquad b_t := \beta \cdot k_t \cdot v_t^\top.$$

Now define an auxiliary sequence $(\tilde{x}_t)_{t\in\mathbb{N}}$ in the ambient state space $\mathbb{R}^2$ by

$$\tilde{x}_0 := (0,0), \qquad \tilde{x}_t \;:=\; A_t \cdot \tilde{x}_{t-1} + b_t \qquad \forall t \in \mathbb{T}.$$

Notice that we do not require $\tilde{x}_t \in X$: $(\tilde{x}_t)_{t\in\mathbb{N}}$ is used only as an auxiliary sequence in the ambient state space.

By its definition, $(\tilde{x}_t)_{t\in\mathbb{N}}$ is exactly the sequence considered in Lemma 41. Hence,

$$\|\tilde{x}_t\| \to \infty \quad \text{as} \quad t \to \infty. \tag{204}$$

We next observe that the state-update matrices A_t are nonexpansive. Indeed, since $\|k_t\| = 1$, every $z \in \mathbb{R}^2$ can be uniquely written as

$$z \;=\; a \cdot k_t + z^\top, \qquad\qquad k_t^\top \cdot z^\top = (0,0).$$

Consequently,

$$A_t \cdot z \;=\; (1-\beta)\cdot a \cdot k_t + z^\top,$$

and therefore

$$\begin{aligned} \|A_t \cdot z\|^2 \;&=\; (1-\beta)^2 \cdot a^2 + \|z^\top\|^2 \\ &\le\; a^2 + \|z^\top\|^2 \\ &=\; \|z\|^2. \end{aligned}$$

Hence,

$$\|A_t\| \le 1 \qquad \forall t \in \mathbb{T}.$$

The sequences $(x_t)_{t\in\mathbb{N}}$ and $(\tilde{x}_t)_{t\in\mathbb{N}}$ therefore satisfy the hypotheses of Lemma 40. Since $\tilde{x}_0 = (0,0)$, that lemma gives

$$\|x_t - \tilde{x}_t\| \le \|x_0\| \qquad \forall t \in \mathbb{N}.$$

Thus, by the reverse triangle inequality,

$$\|x_t\| \ge \|\tilde{x}_t\| - \|x_0\|.$$

Combining this inequality with Equation (204) yields

$$\|x_t\| \to \infty \quad \text{as} \quad t \to \infty.$$

Thus, the bounded admissible input sequence $(u_t)_{t\in\mathbb{T}}$ generates an unbounded state sequence from $x_0 \in X$. Hence D does not satisfy BIBS. □

S.7.2 BIBSD Instability of DeltaNet

Fact 61 (The specialised parametrisation is DeltaNet)**.** *The* specialised parametric DeltaNet Recurrent Unit $D_\star$ *with hyperparameters* $(c, \epsilon, \beta_\star)$ *of Construction 1 is a DeltaNet Recurrent Unit with no decay gate and update hyperparameter c, under the specialised parametrisation*

$$\vartheta \;:\; \lambda \longmapsto \Big(\big(W_{\mathrm{k}}[\lambda], b_{\mathrm{k}}\big),\; \big(W_{\mathrm{v}}, b_{\mathrm{v}}\big),\; \big(w_\beta, b_\beta\big) \Big) \qquad \textit{for} \quad \lambda \in \Lambda_\star\,.$$

Specifically, the standard DeltaNet Recurrent Unit with no decay gate, update hyperparameter c, and parameter space

$$\Theta_\star \;:=\; \mathrm{Im}(\vartheta) \;=\; \big\{\theta[\lambda] : \lambda \in \Lambda_\star\big\}$$

has precisely the dynamics of $D_\star$ *under the parameter identification* ϑ. End of Fact 61.

Proof. We first verify that $\Theta_\star$ is a valid parameter space for a standard DeltaNet Recurrent Unit with no decay gate. Let $\lambda \in \Lambda_\star$ and $u \in U$ be arbitrary. By Construction 1, the key preactivation is

$$\tilde{k}(u;\lambda) \;=\; W_{\mathrm{k}}[\lambda]\cdot u + b_{\mathrm{k}} \;=\; \begin{pmatrix} 1 \\ \lambda u \end{pmatrix}. \tag{205}$$

In particular,

$$\tilde{k}(u;\lambda) \neq 0 \qquad \forall\, \lambda \in \Lambda_\star\,, \quad \forall\, u \in U\,.$$

Hence every $\vartheta(\lambda) \in \Theta_\star$ satisfies the key-admissibility condition of Definition 275. All remaining components of $\vartheta(\lambda)$ have the dimensions required by that definition. Thus, $\Theta_\star$ is a valid parameter space for a DeltaNet Recurrent Unit with no decay gate and update hyperparameter c.

We next compare its dynamics with those of $D_\star$. By Equation (205),

$$k(u;\lambda) \;=\; \frac{\begin{pmatrix}1\\ \lambda u\end{pmatrix}}{\sqrt{1+\lambda^2}}\,. \tag{206}$$

Furthermore, by the parameter choices of Construction 1,

$$\beta(u;\lambda) \;=\; \sigma\left(\log\left(\frac{\beta_\star}{1-\beta_\star}\right)\right) \;=\; \beta_\star\,, \qquad v(u;\lambda) \;=\; u\,.$$

On the other hand, the standard DeltaNet parameter value associated with λ is

$$\vartheta(\lambda) = \Big(\big(W_{\mathrm{k}}[\lambda], b_{\mathrm{k}}\big),\ \big(W_{\mathrm{v}}, b_{\mathrm{v}}\big),\ \big(w_\beta, b_\beta\big)\Big).$$

Therefore, Definition 275 gives, for every $\lambda \in \Lambda_\star$ and $u \in U$, the same key, write strength, and value functions as those of $D_\star$:

$$k\big(u;\vartheta(\lambda)\big) = k(u;\lambda)\,, \qquad \beta\big(u;\vartheta(\lambda)\big) = \beta(u;\lambda)\,, \qquad v\big(u;\vartheta(\lambda)\big) = v(u;\lambda)\,.$$

Consequently, the corresponding state-update matrix and additive term satisfy

$$A\big(u;\vartheta(\lambda)\big) \;=\; A(u;\lambda)\,, \qquad B\big(u;\vartheta(\lambda)\big) \;=\; B(u;\lambda)\,.$$

Hence, for every $H \in X$, $u \in U$, and $\lambda \in \Lambda_\star$,

$$f\big(H,u;\vartheta(\lambda)\big) \;=\; f(H,u;\lambda)\,.$$

Thus, the standard DeltaNet Recurrent Unit with no decay gate, update hyperparameter c, and parameter space $\Theta_\star$ has precisely the dynamics of $D_\star$ under the parameter identification ϑ. □

Lemma 46 (*Aux. for Thms. 58, 59:* Parameter-sensitivity growth of DeltaNet on finite input spaces)**.** *Let $c \in \{1,2\}$, $\epsilon \in \mathbb{R}^{>0}$, and $\beta_\star \in (0,1)$, and let*

$$D_\star = \langle X, U, f; A, B \,|\, \Lambda_\star\rangle$$

be the specialised parametric DeltaNet Recurrent Unit with hyperparameters $(c,\epsilon,\beta_\star)$ as per Construction 1. Let

$$x_0 := (0,0)\,, \qquad u := (u_t)_{t\in\mathbb{T}} \in U^\omega \quad \textit{with} \quad u_t := (-1)^{t+1}\,.$$

For every $t \in \mathbb{T}$, let us define the parameter-to-state function

$$H_t : \Lambda_\star \to \mathbb{R}^2, \qquad H_t(\lambda) := \check{D}_\star[x_0,u,t]\big(\lambda\big)\,,$$

and let $H_t^\star := H_t(\lambda_\star)$ for $\lambda_\star := 0$.

Then the following holds:

- *the state sequence $(H_t^\star)_{t\in\mathbb{T}}$ is bounded:*

$$\sup_{t\in\mathbb{N}} \big\|H_t^\star\big\| < \infty\,; \tag{201}$$

- *for every $t \in \mathbb{T}$, the parameter-to-state function H_t is differentiable at $\lambda_\star$, and it satisfies*

$$\big\|\mathsf{D}H_t(\lambda_\star)\big\| \;\geq\; \beta_\star \cdot t\,; \tag{202}$$

- *consequently,*

$$\big\|\mathsf{D}H_t(\lambda_\star)\big\| \to \infty \quad \textit{as} \quad t \to \infty\,.$$

Proof. We first show the following intermediate claim.

Intermediate Claim. For every $t \in \mathbb{N}$, there exists a scalar sequence $(a_t)_{t\in\mathbb{N}}$ such that, for every $t \in \mathbb{N}$,

$$H_t^\star \;=\; a_t \cdot e_1 \qquad \text{where} \qquad a_t \;=\; \begin{cases} 0 & \text{if} \;\; t = 0\,, \\ (1 - c\cdot\beta_\star)\cdot a_{t-1} + \beta_\star \cdot u_t & \text{if} \;\; t \geq 1\,. \end{cases} \tag{207}$$

Proof of Intermediate Claim. We proceed by induction on $t \in \mathbb{N}$.

In the base case we have $t = 0$, and hence

$$H_0^\star = x_0 = 0 = a_0 \cdot e_1$$

since $x_0 = (0, 0)$ and $a_0 = 0$.

In the inductive case, we have $t \geq 1$, and we assume by indunction that

$$H_{t-1}^\star = a_{t-1} \cdot e_1\,.$$

By the dynamics function of $D_\star$, together with

$$k(u_t; \lambda_\star) = e_1, \qquad \beta(u_t; \lambda_\star) = \beta_\star, \qquad v(u_t; \lambda_\star) = u_t,$$

we obtain

$$\begin{aligned} H_t^\star &= f(H_{t-1}^\star, u_t; \lambda_\star) \\ &= \big(I_2 - c\cdot\beta_\star \cdot e_1 e_1^\top\big)\cdot H_{t-1}^\star + \beta_\star \cdot e_1 \cdot u_t \\ &= \big(I_2 - c\cdot\beta_\star \cdot e_1 e_1^\top\big)\cdot \big(a_{t-1}\cdot e_1\big) + \beta_\star \cdot u_t \cdot e_1 \\ &= \big(a_{t-1} - c\cdot\beta_\star \cdot a_{t-1} \cdot e_1^\top e_1\big)\cdot e_1 + \beta_\star \cdot u_t \cdot e_1 \\ &= \Big((1 - c\cdot\beta_\star)\cdot a_{t-1} + \beta_\star \cdot u_t\Big)\cdot e_1, \end{aligned}$$

where we have used $e_1^\top \cdot e_1 \;=\; 1$. Thus, defining

$$a_t := (1 - c\cdot\beta_\star)\cdot a_{t-1} + \beta_\star \cdot u_t,$$

we have $H_t^\star = a_t \cdot e_1$. This proves the claim by induction. *End of proof of Intermediate Claim.*

Since

$$|1 - c\cdot\beta_\star| \;<\; 1 \qquad \text{and} \qquad |u_t| \;=\; 1 \quad \forall\, t \in \mathbb{T}\,,$$

we have

$$\sup_{t\in\mathbb{T}} \big\|H_t^\star\big\| \;=\; \sup_{t\in\mathbb{T}} |a_t| \;<\; \infty\,,$$

proving Equation (201).

We next consider the derivative with respect to the parameter λ at $\lambda_\star$. We have

$$\mathsf{D}k(u;\cdot)(\lambda_\star) = u \cdot e_2 \qquad \forall\, u \in U\,, \tag{208}$$

and hence,

$$\mathsf{D}\beta(u;\cdot)(\lambda_\star) \;=\; 0 \qquad \text{and} \qquad \mathsf{D}v(u;\cdot)(\lambda_\star) \;=\; 0 \qquad \forall\, u \in U\,.$$

We claim that there exists a scalar sequence $(g_t)_{t\in\mathbb{N}}$ such that

$$\mathsf{D}H_t(\lambda_\star) \;=\; g_t \cdot e_2\,, \qquad g_0 \;=\; 0\,. \tag{209}$$

Indeed, by Construction 1,

$$\begin{aligned} H_t(\lambda) \;=\; &\big(I_2 - c\cdot\beta(u_t;\lambda)\cdot k(u_t;\lambda)\cdot k(u_t;\lambda)^\top\big)\cdot H_{t-1}(\lambda) \\ &+ \beta(u_t;\lambda)\cdot k(u_t;\lambda)\cdot v(u_t;\lambda) \end{aligned} \qquad \forall\, \lambda \in \Lambda_\star\,.$$

Taking the derivative with respect to λ at $\lambda_\star$, and using

$$k(u_t;\lambda_\star) \;=\; e_1, \qquad H^\star_{t-1} \;=\; a_{t-1}\cdot e_1\,, \qquad \mathsf{D}k(u_t;\cdot)(\lambda_\star) \;=\; u_t\cdot e_2\,,$$

gives

$$\begin{aligned}\mathsf{D}H_t(\lambda_\star) \;=\; &\big(I_2 - c\cdot\beta_\star\cdot e_1\cdot e_1^\top\big)\cdot \mathsf{D}H_{t-1}(\lambda_\star)\\ &- c\cdot\beta_\star\cdot\big(u_t\cdot e_2\cdot e_1^\top + u_t\cdot e_1\cdot e_2^\top\big)\cdot a_{t-1}\cdot e_1\\ &+ \beta_\star\cdot u_t\cdot e_2\cdot u_t\\ = &\big(I_2 - c\cdot\beta_\star\cdot e_1\cdot e_1^\top\big)\cdot \mathsf{D}H_{t-1}(\lambda_\star)\\ &- \beta_\star\cdot\big(1 - c\cdot u_t\cdot a_{t-1}\big)\cdot e_2\,.\end{aligned}$$

Since

$$\big(I_2 - c\cdot\beta_\star\cdot e_1\cdot e_1^\top\big)\cdot e_2 \;=\; e_2\,,$$

by induction we obtain Equation (209), with

$$g_t \;=\; g_{t-1} + \beta_\star\cdot\big(1 - c\cdot u_t\cdot a_{t-1}\big)\,. \tag{210}$$

Let $z_t := u_t\cdot a_t$. Since $u_t = -u_{t-1}$, we have

$$z_t \;=\; \beta_\star - (1 - c\cdot\beta_\star)\cdot z_{t-1}\,.$$

Writing

$$\rho \;:=\; 1 - c\cdot\beta_\star \;\in\; (-1,1)\,,$$

and observing that $z_0 = 0$, we obtain

$$z_t \;=\; \beta_\star\cdot\sum_{j=0}^{t-1}(-\rho)^j \;=\; \beta_\star\cdot\frac{1-(-\rho)^t}{1+\rho} \;>\; 0 \qquad \forall\, t\in\mathbb{T}\,.$$

Therefore, for every $t \geq 2$,

$$u_t\cdot a_{t-1} \;=\; -z_{t-1} \;\leq\; 0\,.$$

Substituting this into Equation (210) gives

$$g_t \;\geq\; g_{t-1} + \beta_\star \qquad \forall\, t\geq 2\,.$$

Furthermore, Equation (210), together with $a_0 = 0$ and $g_0 = 0$, gives

$$g_1 \;=\; \beta_\star\,.$$

Hence, by induction we obtain

$$g_t \;\geq\; \beta_\star\cdot t \qquad \forall\, t\in\mathbb{N}\,.$$

Combining this with Equation (209), we obtain

$$\big\|\mathsf{D}H_t(\lambda_\star)\big\| \;=\; |g_t| \;\geq\; \beta_\star\cdot t \qquad \forall\, t\in\mathbb{N}\,,$$

proving Equation (202).

The divergence claim follows immediately. □

Theorem 62 (DeltaNet does not satisfy wBIBSD)**.** *Let $c\in\{1,2\}$, $\epsilon\in\mathbb{R}^{>0}$, and $\beta_\star\in(0,1)$.*

Let

$$D_\star = \langle X, U, f; A, B \,|\, \Lambda_\star\rangle$$

be the specialised parametric DeltaNet Recurrent Unit with hyperparameters $(c,\epsilon,\beta_\star)$ of Construction 1.

Then the DeltaNet Unit $D_\star$ does not satisfy wBIBSD, and consequently it does not satisfy uBIBSD.

End of Theorem 62.

Proof. Let

$$x_0 := (0,0) \qquad \text{and} \qquad u := (u_t)_{t\in\mathbb{T}} \in U^\omega \quad \text{with} \quad u_t := (-1)^{t+1}.$$

Let moreover

$$\lambda_\star := 0.$$

Taking $r := 1$, we have

$$u \in U^\omega_{\leq r}, \qquad x_0 \in X_{\leq r}, \qquad \lambda_\star \in (\Lambda_\star)_{\leq r}.$$

By Lemma 42, for every $t \in \mathbb{T}$, the parameter-to-state function

$$H_t := \check{D}_\star[x_0, u, t]$$

is differentiable at $\lambda_\star$ and satisfies

$$\left\|\mathsf{D}H_t(\lambda_\star)\right\| \geq \beta_\star \cdot t.$$

Hence

$$\sup_{t\in\mathbb{T}} \left\|\mathsf{D}\check{D}_\star[x_0, u, t](\lambda_\star)\right\| = \infty.$$

Therefore, there exists no constant $B_{r,u,x_0} \in \mathbb{R}^{\geq 0}$ satisfying the derivative bound required by Definition 180 for this r, u, and x_0. Thus, $D_\star$ does not satisfy wBIBSD.

Since uBIBSD implies wBIBSD, $D_\star$ does not satisfy uBIBSD either. □

S.7.3 BIBLG Instability of DeltaNet

Theorem 63 (Linear loss-gradient growth of DeltaNet (already on finite input spaces))**.** *Let $c \in \{1,2\}$, let $\beta_\star \in (0,1)$, and let $\ell : Y \times \hat{Y} \to \mathbb{R}$ be an analytic loss function with $Y := \mathbb{R}^d$ and $\hat{Y} \subseteq \mathbb{R}^{d_{\text{target}}}$.*

Suppose that ℓ satisfies the nonzero local derivative condition as witnessed by $(a, y_\star, \hat{y}_\star)$.

For some $\epsilon \in \mathbb{R}^{>0}$, a DeltaNet model $S \in \mathcal{M}[\mathrm{DeltaNet}|c]$, with input space U, output space $Y = \mathbb{R}^d$, parameter space Θ satisfying

$$U_\star := \{-1, 1\} \subseteq U\,, \qquad \Theta_\star \subseteq \Theta\,,$$

where $\Theta_\star$ is as per Construction 1 under the present $\beta_\star$ and ϵ, for which there exist a radius $r \in \mathbb{R}^{>0}$, an input sequence $u \in U^\omega_\star$, a parameter $\theta_\star \in \Theta_\star$, and a constant $C \in \mathbb{R}^{>0}$ such that, for every $T \in \mathbb{N}^{>0}$, we have that

$$z_T := (u_{[1..T]}, \hat{y}_\star)$$

is an admissible sample for S, and additionally:

(1) the sample z_T and parameter $\theta_\star$ belong to their corresponding truncated sets:

$$z_T \in \mathcal{Z}^{S,\leq T}_{\leq r}\,, \qquad \theta_\star \in \Theta_{\leq r}\,;$$

(2) the objective function $\mathcal{L}^\ell_{S,z_T}$ is differentiable at $\theta_\star$;

(3) the loss gradient is bounded from below as follows:

$$\left\|\mathsf{D}\mathcal{L}^\ell_{S,z_T}(\theta_\star)\right\| \geq C \cdot T\,. \tag{193}$$

Consequently, for r the above radius, the growth of the corresponding loss-gradients norm is asymptotically lower-bounded as

$$\mathsf{G}^{\mathrm{S},\ell,\mathrm{r}}_{\nabla\mathcal{L}}(T) \;=\; \Omega(T)\,.$$

In particular, S does not satisfy BIBLG, and hence $\mathcal{M}[\mathrm{DeltaNet}|c]$ does not satisfy BIBLG, even when the input space is restricted to a fixed two-symbol alphabet. End of Theorem 63.

Proof. Apply Lemma 42 for the present values of c and $\beta_\star$. Let $\epsilon \in \mathbb{R}^{>0}$,

$$D = \langle X, U, f; A, B \,|\, \Theta \rangle,$$

and

$$u = (u_t)_{t \in \mathbb{N}^{>0}} \in U_\star^\omega, \qquad U_\star := \{-1, 1\},$$

be witnesses supplied by that lemma.

Let $\lambda_\star$, $\Lambda_\star$, $\Theta_\star$, and $\theta[\lambda]$ be as in Construction 1 for the present values of ϵ and $\beta_\star$. Thus,

$$\lambda_\star = 0, \qquad \Theta_\star \subseteq \Theta.$$

Define

$$\theta_\star := \theta[\lambda_\star]. \tag{211}$$

Since $\lambda_\star \in \Lambda_\star$, we have

$$\theta_\star \in \Theta_\star \subseteq \Theta.$$

Let

$$x_0 := (0, 0).$$

By Lemma 42, the reference state trajectory

$$H_T^\star := \chi_T^{u, x_0}(\theta_\star)$$

is bounded:

$$\sup_{T \in \mathbb{N}} \|H_T^\star\| < \infty. \tag{212}$$

We next record the form of the reference states. By Construction 1, for every $\xi \in U$,

$$k(\xi; \theta_\star) = e_1.$$

Hence, by induction on T, the reference state has the form

$$H_T^\star = a_T e_1 \tag{213}$$

for some scalar sequence $(a_T)_{T \in \mathbb{N}}$. By Equation (212), this scalar sequence is bounded.

We also record the direction of the parameter sensitivity. Let

$$G_T := \mathsf{D}_{\lambda_\star} \chi_T^{u, x_0}(\theta_\star).$$

By the explicit parametrisation in Construction 1, for every $\xi \in U$,

$$\mathsf{D}_{\lambda_\star} k(\xi; \cdot)(\theta_\star) = \xi e_2,$$

whereas β and v are constant with respect to this scalar parameter coordinate.

Differentiating the DeltaNet recurrence at $\theta_\star$ therefore gives

$$G_T = \big(I_2 - c\beta_\star e_1 e_1^\top\big)\, G_{T-1} + \beta_\star \big(1 - c u_T a_{T-1}\big) e_2.$$

Since

$$G_0 = 0$$

and

$$\big(I_2 - c\beta_\star e_1 e_1^\top\big)\, e_2 = e_2,$$

induction yields

$$G_T = g_T e_2$$

for some scalar sequence $(g_T)_{T \in \mathbb{N}}$. Consequently, Equation (202) gives

$$\mathsf{D}_{\lambda_\star} \chi_T^{u, x_0}(\theta_\star) = g_T e_2, \qquad |g_T| \geq \beta_\star \cdot T \qquad \forall T \in \mathbb{N}^{>0}. \tag{214}$$

Let S be the one-unit DeltaNet model obtained from D, with initial state

$$H^{\mathrm{init}} := x_0,$$

output space

$$Y = \mathbb{R}^d,$$

associated target space $\hat{Y}$, and output function

$$h : X \times U \times X \to \mathbb{R}^d, \qquad h(H, v, H') := y_\star + \left(e_2^\top H'\right) a. \tag{215}$$

The function h is affine and hence analytic. By construction,

$$S \in \mathcal{M}[\mathrm{DeltaNet}|c], \qquad U_\star \subseteq U, \qquad \Theta_\star \subseteq \Theta.$$

Since $\ell : Y \times \hat{Y} \to \mathbb{R}$ is analytic, it is an admissible loss function for S.

For every $T \in \mathbb{N}^{>0}$, let

$$R_T : \Theta \to \mathbb{R}^d, \qquad R_T(\theta) := \check{S}(u_{[1..T]}; \theta)$$

be the parameter-to-representation map corresponding to the length-T prefix of u. By Equations (213) and (215),

$$R_T(\theta_\star) = y_\star \qquad \forall T \in \mathbb{N}^{>0}, \tag{216}$$

since

$$e_2^\top H_T^\star = 0.$$

By the definition of R_T, Equation (215), and Equation (214), its named partial derivative with respect to the scalar parameter coordinate whose value at $\theta_\star$ is $\lambda_\star$ satisfies

$$\begin{aligned} \mathsf{D}_{\lambda_\star} R_T(\theta_\star) &= \left(e_2^\top \mathsf{D}_{\lambda_\star} \chi_T^{u,x_0}(\theta_\star)\right) a \\ &= g_T a. \end{aligned} \tag{217}$$

For every $T \in \mathbb{N}^{>0}$, define

$$z_T := (u_{[1..T]}, \hat{y}_\star). \tag{218}$$

Since

$$u_{[1..T]} \in U^+ \qquad \text{and} \qquad \hat{y}_\star \in \hat{Y},$$

we have

$$z_T \in \mathcal{Z}^S,$$

so z_T is an admissible sample for S.

We next verify differentiability of its objective function. By the key-admissibility condition of Definition 275, for every $\theta \in \Theta$ and $v \in U$,

$$\widetilde{k}(v; \theta) \neq 0.$$

Hence the normalisation map

$$q \longmapsto \frac{q}{\|q\|}$$

is smooth at every key preactivation arising from an admissible input-parameter pair. All remaining recurrence-input functions of DeltaNet are compositions of affine maps and smooth functions. Therefore, for every finite input sequence, the corresponding state and representation maps are differentiable with respect to the parameters. Since h is affine and ℓ is analytic, it follows that

$$\mathcal{L}^\ell_{S,z_T}$$

is differentiable at $\theta_\star$ for every $T \in \mathbb{N}^{>0}$.

Applying the chain rule to the named partial derivative of $\mathcal{L}^\ell_{S,z_T}$ with respect to the scalar parameter coordinate whose value at $\theta_\star$ is $\lambda_\star$, and using Equations (216) and (217), yields

$$\begin{aligned} \mathsf{D}_{\lambda_\star} \mathcal{L}^\ell_{S,z_T}(\theta_\star) &= \mathsf{D}_{y_\star} \ell\left(y_\star, \hat{y}_\star\right) \cdot \mathsf{D}_{\lambda_\star} R_T(\theta_\star) \\ &= g_T \cdot \left(\mathsf{D}_{y_\star} \ell\left(y_\star, \hat{y}_\star\right) \cdot a\right). \end{aligned} \tag{219}$$

Define

$$\eta := \left|\mathsf{D}_{y_\star} \ell\left(y_\star, \hat{y}_\star\right) \cdot a\right|.$$

By the nonzero local derivative condition,

$$\eta > 0.$$

Therefore, by Equations (214) and (219),

$$\left|\mathsf{D}_{\lambda_\star}\mathcal{L}^{\ell}_{S,z_T}(\theta_\star)\right| \geq \eta\beta_\star T \qquad \forall T \in \mathbb{N}^{>0}. \tag{220}$$

Since λ is a scalar parameter coordinate and its value at $\theta_\star$ is $\lambda_\star$,

$$\mathsf{D}_{\lambda_\star}\mathcal{L}^{\ell}_{S,z_T}(\theta_\star)$$

is the corresponding scalar coordinate of

$$\mathsf{D}\mathcal{L}^{\ell}_{S,z_T}(\theta_\star).$$

Consequently, Equation (220) implies

$$\begin{aligned}\left\|\mathsf{D}\mathcal{L}^{\ell}_{S,z_T}(\theta_\star)\right\| &\geq \left|\mathsf{D}_{\lambda_\star}\mathcal{L}^{\ell}_{S,z_T}(\theta_\star)\right| \\ &\geq \eta\beta_\star T.\end{aligned} \tag{221}$$

Hence Equation (193) holds with

$$C := \eta\beta_\star \in \mathbb{R}^{>0}.$$

It remains to choose a common truncation radius. Since $U_\star$ is finite, and $\theta_\star$ and $\hat{y}_\star$ are fixed finite-dimensional elements, there exists $r \in \mathbb{R}^{>0}$ such that

$$U_\star \subseteq U_{\leq r}, \qquad \theta_\star \in \Theta_{\leq r}, \qquad \hat{y}_\star \in \hat{Y}_{\leq r}. \tag{222}$$

Since

$$u \in U_\star^{\omega},$$

every input appearing in $u_{[1..T]}$ belongs to $U_{\leq r}$. Together with Equation (222), this gives

$$z_T \in \mathcal{Z}_{\leq r}^{S,\leq T} \qquad \forall T \in \mathbb{N}^{>0}. \tag{223}$$

Thus all three claims in the first part of the theorem hold for this single radius r.

By the definition of the corresponding loss-gradient growth function, Equations (222) and (223) imply

$$\begin{aligned}\mathsf{G}^{\mathrm{S},\ell,\mathrm{r}}_{\nabla\mathcal{L}}(T) &= \sup\left\{\left\|\mathsf{D}\mathcal{L}^{\ell}_{S,z}(\theta)\right\| \;:\; z \in \mathcal{Z}_{\leq r}^{S,\leq T},\ \theta \in \Theta_{\leq r}\right\} \\ &\geq \left\|\mathsf{D}\mathcal{L}^{\ell}_{S,z_T}(\theta_\star)\right\|.\end{aligned}$$

Hence, by Equation (221),

$$\mathsf{G}^{\mathrm{S},\ell,\mathrm{r}}_{\nabla\mathcal{L}}(T) \geq CT \qquad \forall T \in \mathbb{N}^{>0}.$$

Therefore,

$$\mathsf{G}^{\mathrm{S},\ell,\mathrm{r}}_{\nabla\mathcal{L}}(T) = \Omega(T).$$

In particular, the loss-gradient norm is not uniformly bounded over the fixed-radius truncated sample and parameter sets as the horizon grows. By Definition 191, S therefore does not satisfy BIBLG. Since

$$S \in \mathcal{M}[\mathrm{DeltaNet}|c],$$

the class $\mathcal{M}[\mathrm{DeltaNet}|c]$ does not satisfy BIBLG either. Finally, the witnessing input sequence takes values entirely in the fixed two-symbol set

$$U_\star = \{-1, 1\},$$

proving the final claim. □

Lemma 47 (Standard losses satisfy the nonzero local derivative condition)**.** *Each of the standard loss functions is analytic and hence admissible for every system having matching output space. Furthermore, each of the standard loss functions satisfies the nonzero local derivative condition:*[22]

(1) the squared-error loss $\ell_{\mathrm{SE}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ satisfies the nonzero local derivative condition as witnessed by $(a_{\mathrm{SE}},\, y_{\mathrm{SE}},\, \hat{y}_{\mathrm{SE}}) := (e_1, e_1, 0)$, and specifically

$$\mathsf{D}_{y_{\mathrm{SE}}}\ell_{\mathrm{SE}}\big(y_{\mathrm{SE}},\, \hat{y}_{\mathrm{SE}}\big) \cdot a_{\mathrm{SE}} \;=\; 1\,; \tag{194}$$

[22] The *nonzero local derivative condition* is introduced in Definition 276.

(2) the binary cross-entropy loss $\ell_{\text{BCE}} : \mathbb{R} \times \{0,1\} \to \mathbb{R}$ satisfies the nonzero local derivative condition as witnessed by $(a_{\text{BCE}},\, y_{\text{BCE}},\, \hat{y}_{\text{BCE}}) := (1,0,0)$, and specifically

$$\mathsf{D}_{y_{\text{BCE}}}\ell_{\text{BCE}}\big(y_{\text{BCE}},\, \hat{y}_{\text{BCE}}\big) \cdot a_{\text{BCE}} \;=\; \frac{1}{2}\,; \tag{195}$$

(3) for every $K \in \mathbb{N}^{>0}$ with $K \geq 2$, the K-class cross-entropy loss $\ell_{\text{CE-}K} : \mathbb{R}^K \times [1..K] \to \mathbb{R}$ satisfies the nonzero local derivative condition as witnessed by $(a_{\text{CE}},\, y_{\text{CE}},\, \hat{y}_{\text{CE}}) := (e_2, 0, 1)$, and specifically

$$\mathsf{D}_{y_{\text{CE}}}\ell_{\text{CE-}K}\big(y_{\text{CE}},\, \hat{y}_{\text{CE}}\big) \cdot a_{\text{CE}} \;=\; \frac{1}{K}\,. \tag{196}$$

End of Lemma 47.

Proof. We consider the three standard loss functions separately.

Squared error. The squared-error loss

$$\ell_{\text{SE}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{\geq 0}\,, \qquad\qquad \ell_{\text{SE}}(y,\hat{y}) := \frac{1}{2}\|y-\hat{y}\|^2\,,$$

is polynomial and hence analytic. Therefore, it is admissible for every system with matching output space $Y = \mathbb{R}^d$.

Its partial derivative with respect to its input block y_{SE} evaluated at $(y_{\text{SE}}, \hat{y}_{\text{SE}}) = (e_1, 0)$ is

$$\mathsf{D}_{y_{\text{SE}}}\ell_{\text{SE}}\big(y_{\text{SE}}, \hat{y}_{\text{SE}}\big) \;=\; \big(y_{\text{SE}} - \hat{y}_{\text{SE}}\big)^\top \;=\; e_1^\top\,.$$

Since $a_{\text{SE}} = e_1$, it follows that

$$\mathsf{D}_{y_{\text{SE}}}\ell_{\text{SE}}\big(y_{\text{SE}}, \hat{y}_{\text{SE}}\big) \cdot a_{\text{SE}} \;=\; e_1^\top \cdot e_1 \;=\; 1\,.$$

This proves Equation (194), and hence that $(a_{\text{SE}}, y_{\text{SE}}, \hat{y}_{\text{SE}})$ witnesses the nonzero local derivative condition for ℓ_{SE}.

Binary cross-entropy. The function

$$\overline{\ell_{\text{BCE}}} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, \qquad\qquad \overline{\ell_{\text{BCE}}}(y,\hat{y}) := \log\big(1+\exp(y)\big) - \hat{y}\cdot y$$

is analytic and is an ambient extension of ℓ_{BCE}, showing that ℓ_{BCE} is analytic and hence admissible for every system with matching output space $Y = \mathbb{R}$.

For the fixed target $\hat{y}_{\text{BCE}} = 0$, the function

$$y \longmapsto \ell_{\text{BCE}}\big(y, \hat{y}_{\text{BCE}}\big)$$

is the analytic function

$$y \longmapsto \log\big(1+\exp(y)\big) - \hat{y}_{\text{BCE}}\cdot y\,.$$

Hence its derivative at $y_{\text{BCE}} = 0$ gives

$$\mathsf{D}_{y_{\text{BCE}}}\ell_{\text{BCE}}\big(y_{\text{BCE}}, \hat{y}_{\text{BCE}}\big) \;=\; \frac{\exp(y_{\text{BCE}})}{1+\exp(y_{\text{BCE}})} - \hat{y}_{\text{BCE}} \;=\; \sigma(y_{\text{BCE}}) - \hat{y}_{\text{BCE}}\,.$$

Since $y_{\text{BCE}} = \hat{y}_{\text{BCE}} = 0$ and $a_{\text{BCE}} = 1$, it follows that

$$\mathsf{D}_{y_{\text{BCE}}}\ell_{\text{BCE}}\big(y_{\text{BCE}}, \hat{y}_{\text{BCE}}\big) \cdot a_{\text{BCE}} \;=\; \sigma(0) \;=\; \frac{1}{2}\,.$$

This proves Equation (195), and hence that $(a_{\text{BCE}}, y_{\text{BCE}}, \hat{y}_{\text{BCE}})$ witnesses the nonzero local derivative condition for ℓ_{BCE}.

K-class cross-entropy. Let $K \in \mathbb{N}^{>0}$ with $K \geq 2$. For every $k \in [1..K]$, define the polynomial

$$p_k(t) \;:=\; \prod_{\substack{m=1 \\ m\neq k}}^{K} \frac{t-m}{k-m}\,.$$

By the *Lagrange interpolation theorem*—cf. Theorem 6.1 of [75]—we have

$$p_j(k) = \begin{cases} 1 & \text{if } j = k, \\ 0 & \text{if } j \neq k, \end{cases} \qquad \forall j, k \in [1..K].$$

Consider the function

$$\overline{\ell_{\text{CE-}K}} : \mathbb{R}^K \times \mathbb{R} \to \mathbb{R}, \qquad \overline{\ell_{\text{CE-}K}}(y, t) := \log\left(\sum_{j=1}^{K} \exp(y_j)\right) - \left(\sum_{j=1}^{K} p_j(t) \cdot y_j\right).$$

This function is analytic. Furthermore, for every $\hat{y} \in [1..K]$,

$$\overline{\ell_{\text{CE-}K}}(y, \hat{y}) \;=\; \log\left(\sum_{j=1}^{K} \exp(y_j)\right) - y_{\hat{y}} \;=\; -\log\left(\frac{\exp(y_{\hat{y}})}{\sum_{j=1}^{K} \exp(y_j)}\right) \;=\; \ell_{\text{CE-}K}(y, \hat{y}) \,.$$

Hence $\overline{\ell_{\text{CE-}K}}$ is an analytic ambient extension of $\ell_{\text{CE-}K}$, showing that $\ell_{\text{CE-}K}$ is analytic and hence admissible for every system with matching output space $Y = \mathbb{R}^K$.

For the fixed target $\hat{y}_{\text{CE}} = 1$, the function

$$y \longmapsto \ell_{\text{CE-}K}\big(y, \hat{y}_{\text{CE}}\big)$$

is the analytic function

$$y \longmapsto \log\left(\sum_{j=1}^{K} \exp(y_j)\right) - y_{\hat{y}_{\text{CE}}} \,.$$

Hence its partial derivative with respect to its input block y_{CE} evaluated at $(y_{\text{CE}}, \hat{y}_{\text{CE}}) = (0, 1)$ is

$$\mathsf{D}_{y_{\text{CE}}} \ell_{\text{CE-}K}\big(y_{\text{CE}}, \hat{y}_{\text{CE}}\big) = \frac{1}{K} \cdot \mathbf{1}_K^\top - e_1^\top \,.$$

Since $a_{\text{CE}} = e_2$ and $K \geq 2$, it follows that

$$\mathsf{D}_{y_{\text{CE}}} \ell_{\text{CE-}K}\big(y_{\text{CE}}, \hat{y}_{\text{CE}}\big) \cdot a_{\text{CE}} \;=\; \left(\frac{1}{K}\mathbf{1}_K^\top - e_1^\top\right) \cdot e_2 \;=\; \frac{1}{K} \,.$$

This proves Equation (196), and hence that $(a_{\text{CE}}, y_{\text{CE}}, \hat{y}_{\text{CE}})$ witnesses the nonzero local derivative condition for $\ell_{\text{CE-}K}$. □

Theorem 64 (Finite-alphabet loss-gradient growth for standard loss functions)**.** *Let $c \in \{1, 2\}$, let $\beta_\star \in (0, 1)$, and let $U_\star := \{-1, 1\}$. For each of the standard loss functions of Definition 277, there exists a DeltaNet model $S \in \mathcal{M}[\text{DeltaNet}|c]$ having input space $U_\star \subseteq U$ and admitting*

$$\textit{radius } r \in \mathbb{R}^{>0}, \qquad \textit{parameters } \theta_\star \in \Theta_{\leq r}, \qquad \textit{sample } z_T \in \mathcal{Z}_{\leq r}^{S,\leq T} \textit{ for each } T \in \mathbb{T},$$

such that:

$$\left\|\mathsf{D}\mathcal{L}_{S,z_T}^{\ell_{\text{SE}}}(\theta_\star)\right\| \;\geq\; \beta_\star \cdot T \qquad \textit{for squared error,} \tag{197}$$

$$\left\|\mathsf{D}\mathcal{L}_{S,z_T}^{\ell_{\text{BCE}}}(\theta_\star)\right\| \;\geq\; \frac{\beta_\star}{2} \cdot T \qquad \textit{for binary cross-entropy,} \tag{198}$$

$$\left\|\mathsf{D}\mathcal{L}_{S,z_T}^{\ell_{\text{CE-}K}}(\theta_\star)\right\| \;\geq\; \frac{\beta_\star}{K} \cdot T \qquad \textit{for } K\textit{-class cross-entropy, } \forall K \geq 2. \tag{199}$$

Consequently,

- *the loss-gradient growth is*

$$\mathsf{G}_{\nabla\mathcal{L}}^{\text{S},\ell,\text{r}}(T) = \Omega(T) \qquad \forall \ell \in \{\ell_{\text{SE}}, \ell_{\text{BCE}}\} \cup \{\ell_{\text{CE-}K}\}_{K \geq 2},$$

- *the class $\mathcal{M}[\text{DeltaNet}|c]$ does not satisfy BIBLG.*

Proof. By Lemma 39, the squared-error, binary cross-entropy, and K-class cross-entropy losses are admissible and satisfy the nonzero local derivative condition, with respective witness derivative magnitudes

$$1, \qquad \frac{1}{2}, \qquad \frac{1}{K}.$$

Therefore, Theorem 59 applies in each of the three cases and yields respectively

$$C = \beta_\star, \qquad C = \frac{\beta_\star}{2}, \qquad C = \frac{\beta_\star}{K}.$$

Hence Equations (197)–(199) hold. The asymptotic loss-gradient-growth and BIBLG claims follow directly from the same theorem. □

T Analysis of Mamba

Table of Contents

T.1 Definition of Mamba 326
T.2 Mamba models are smooth unitwise-contractive 327
T.3 Positive-margin expressivity of Mamba 329

T.1 Definition of Mamba

Definition 278 (Mamba, cf. [13])**.** Let $d_{\mathrm{in}}, d_{\mathrm{s}}, n_{\mathrm{h}} \in \mathbb{N}^{>0}$. A *Mamba Recurrent Unit* of *state dimension* d_{s} and *number of heads* n_{h} is a Parametric General Linear Recurrent Unit

$$D = \langle X, U, f; \Theta \,|\, A, B \rangle$$

with components as described next.

▸ **SubDefinition 278.1** (Input and state spaces). The input and parameter spaces satisfy

$$U \subseteq \Omega_U := \mathbb{R}^{d_{\mathrm{in}}} \qquad \text{and} \qquad X \subseteq \Omega_X := \mathbb{R}^{d_{\mathrm{s}} \times n_{\mathrm{h}}}.$$

End of SD.278.1 ◂

▸ **SubDefinition 278.2** (Parameter space). The parameter space satisfies $\Theta \subseteq \Omega_\Theta$ for Ω_Θ consisting of each parameter value θ of the form

$$\theta \;=\; \big(a,\, (W_{\mathrm{v}}, b_{\mathrm{v}}),\, (W_{\mathrm{b}}, b_{\mathrm{b}}),\, (W_\Delta, b_\Delta)\big) \tag{224}$$

with components

$a \in \mathbb{R}^{d_{\mathrm{s}} \times n_{\mathrm{h}}}$		*decay parameters*
$W_{\mathrm{v}} \in \mathbb{R}^{n_{\mathrm{h}} \times d_{\mathrm{in}}}$	$b_{\mathrm{v}} \in \mathbb{R}^{n_{\mathrm{h}}}$	*value parameters*
$W_{\mathrm{b}} \in \mathbb{R}^{d_{\mathrm{s}} \times d_{\mathrm{in}}}$	$b_{\mathrm{b}} \in \mathbb{R}^{d_{\mathrm{s}}}$	*state-input parameters*
$W_\Delta \in \mathbb{R}^{n_{\mathrm{h}} \times d_{\mathrm{in}}}$	$b_\Delta \in \mathbb{R}^{n_{\mathrm{h}}}$	*step-size parameters*.

End of SD.278.2 ◂

▸ **SubDefinition 278.3** (Recurrence-input functions). The recurrence inputs are given by the parametric functions

$\Delta : U \times \Theta \to (\mathbb{R}^{>0})^{n_{\mathrm{h}}}$	$\Delta(u;\theta) := \mathrm{softplus}\big(\tilde{\Delta}(u;\theta)\big)$	*step size*
$\tilde{\Delta} : U \times \Theta \to \mathbb{R}^{n_{\mathrm{h}}}$	$\tilde{\Delta}(u;\theta) := W_\Delta \cdot u + b_\Delta$	*step-size preactivation*
$v : U \times \Theta \to \mathbb{R}^{n_{\mathrm{h}}}$	$v(u;\theta) := W_{\mathrm{v}} \cdot u + b_{\mathrm{v}}$	*state-input values*
$b : U \times \Theta \to \mathbb{R}^{d_{\mathrm{s}}}$	$b(u;\theta) := W_{\mathrm{b}} \cdot u + b_{\mathrm{b}}$	*state-input vector*

where θ is of the form (224), and the function softplus is applied coordinatewise.

End of SD.278.3 ◂

▸ **SubDefinition 278.4** (State-update coefficients). The *state-update coefficients* are given by the parametric function

$$A : U \times \Theta \to (0,1)^{d_{\mathrm{s}} \times n_{\mathrm{h}}} \qquad A(u;\theta) := \exp\Big(-\exp(a) \cdot \mathrm{Diag}\big(\Delta(u;\theta)\big)\Big)$$

where θ is of the form (224), and the function exp is applied coordinatewise. *End of SD.278.4* ◂

▸ **SubDefinition 278.5** (Dynamics function). The parametric dynamics function is

$$f(X, u; \theta) := \Lambda_u(X;\theta) + B(u;\theta)$$

where the state-update linear operator and additive term are given by the parametric functions

$$\Lambda_u(X;\theta) := A(u;\theta) * X \qquad \textit{state-update linear operator}$$
$$B(u;\theta) := b(u;\theta) \cdot \big(\Delta(u;\theta) \odot v(u;\theta)\big)^\top \qquad \textit{additive term}$$

where $*$ is element-wise product. *End of SD.278.5* ◂

▸ **SubDefinition 278.6**. Next we provide definitions that build on the intermediate definitions introduced above.

The *class of Mamba Recurrent Units* is denoted by $\mathcal{U}[\mathrm{Mamba}]$.

A *Mamba model* is a parametric $\mathcal{U}[\mathrm{Mamba}]$-RNC with initial states set to zero.

The *class of Mamba models* is denoted by $\mathcal{M}[\mathrm{Mamba}]$.

The *class of Mamba dynamics* is the set conssisting of dynamics of each Mamba model,

$$\mathcal{D}[\mathrm{Mamba}] := \big\{\mathrm{dyn}(S) \mid S \in \mathcal{M}[\mathrm{Mamba}]\big\}.$$

End of SD.278.6 ◂

End of Definition 278.

T.2 Mamba models are smooth unitwise-contractive

We show that Mamba models are contractive, as an intermediate result to show the expressivity expressivity upper bound of Theorem 66.

Recalling the notion of *smooth unitwise-contractivity* introduced in Definition 273, we show next that Mamba models are smooth unitwise-contractive. We note that smooth unitwise-contractivity is defined for Linear RNNs, that are RNCs of Linear Recurrent Units, whereas Mamba models are RNCs of Generalised Linear Recurrent Units; thus, the result below is developed for the canonical Linear RNN associated with a Mamba model, as introduced in Definition 268.

Theorem 65 (Presentation-induced Linear RNN representations of Mamba models are smooth contractive)**.** *Let $S \in \mathcal{M}[\mathrm{Mamba}]$, and let*

$$\mathsf{S} := \langle \mathsf{C}, x^{\mathrm{init}} \rangle \triangleright h$$

be a parametric analytic output extension presentation of S such that C is a parametric recurrent cascade presentation of type $(\mathcal{U}[\mathrm{Mamba}], \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}})$.

Let $\tilde{\mathsf{S}}$ be the Linear RNN presentation associated with S according to Definition 268, and let $\tilde{S} := [\![\tilde{\mathsf{S}}]\!]$ be the corresponding Linear RNN representation of S.

Then $\tilde{S}$ is a smooth unitwise-contractive Linear RNN. End of Theorem 65.

Proof. For each $i \in [1..\ell]$, let us write

$$\mathsf{C} \;=\; g_1 \triangleright \mathsf{L}_1 \blacktriangleleft_{g_2} \mathsf{L}_2 \blacktriangleleft_{g_3} \cdots \blacktriangleleft_{g_\ell} \mathsf{L}_\ell, \qquad \text{where} \qquad \mathsf{L}_i \;=\; D_{i,1} \parallel \cdots \parallel D_{i,m_i}\,.$$

Since C is a parametric recurrent cascade presentation of type $(\mathcal{U}[\mathrm{Mamba}], \mathcal{F}^{\mathrm{p}}_{\mathrm{AN}})$, we have that every $D_{i,j}$ is a Mamba Recurrent Unit and every parametric connecting function g_i is analytic.

Furthermore, every Mamba Recurrent Unit is a Parametric General Linear Recurrent Unit by Definition 278. Hence Proposition 65 applies to S, and therefore $\tilde{S}$ is a Parametric Linear RNN.

It remains to verify the additional requirements of Definition 273.

Let us fix arbitrary $i \in [1..\ell]$ and $j \in [1..m_i]$, and let us write the corresponding Mamba Recurrent Unit as

$$D_{i,j} \;=\; \langle X, U, f; \Theta \,|\, A, B\rangle, \qquad \text{with} \quad X \subseteq \mathbb{R}^{d_{\mathrm{s}} \times n_{\mathrm{h}}}.$$

Let

$$\tilde{D}_{i,j} \;=\; \langle \tilde{X}, U, \tilde{f}; \Theta \,|\, \tilde{A}, \tilde{B}\rangle$$

be its canonical Linear Recurrent Unit representation, part of the flattened layer presentation for layer i under the layer presentation $\tilde{\mathsf{L}}_i$.

By Definition 278, the state-update coefficient matrix is

$$A(u;\theta) \;=\; \exp\Big(-\exp(a) \cdot \mathrm{Diag}\big(\Delta(u;\theta)\big)\Big), \tag{225}$$

where

$$\Delta(u;\theta) \;=\; \mathrm{softplus}\big(W_\Delta \cdot u + b_\Delta\big), \tag{226}$$

and the state-update linear operator satisfies

$$\Lambda_u(X;\theta) \;=\; A(u;\theta) * X. \tag{227}$$

For every $X \in \mathbb{R}^{d_\mathrm{s}\times n_\mathrm{h}}$, Equation (227) gives

$$\begin{aligned}\mathrm{flatten}\left(\Lambda_u(X;\theta)\right) &= \mathrm{flatten}\left(A(u;\theta) * X\right)\\ &= \mathrm{Diag}\left(\mathrm{flatten}\big(A(u;\theta)\big)\right)\cdot \mathrm{flatten}(X).\end{aligned}$$

Therefore, by Definition 265, the matrix representation under flattening of the state-update linear operator is

$$\mathrm{flatten}\left(\Lambda_u(\cdot;\theta)\right) \;=\; \mathrm{Diag}\left(\mathrm{flatten}\big(A(u;\theta)\big)\right).$$

Hence, by Definition 267, the state-update matrix of $\tilde{D}_{i,j}$ is

$$\tilde{A}(u;\theta) \;=\; \mathrm{Diag}\left(\mathrm{flatten}\big(A(u;\theta)\big)\right), \tag{228}$$

and its additive term is

$$\tilde{B}(u;\theta) \;=\; \mathrm{flatten}\big(B(u;\theta)\big). \tag{229}$$

Continuous differentiability up to the closure. The functions Δ, A, and B of Definition 278 are obtained from affine maps, $\exp$, $\mathrm{softplus}$, linear maps such as Diag, and finite-dimensional algebraic operations by finite tupling and composition. Hence, by Proposition 2 and Proposition 1, they are analytic Euclidean maps.

Since $\mathrm{flatten}$ and Diag are linear maps, Equations (228) and (229) imply that $\tilde{A}$ and $\tilde{B}$ are analytic Euclidean maps as well. By Fact 11, they are continuously-differentiable up to the closures of their domains.

Strict contractivity on the closure. Let $(\overline{u},\overline{\theta}) \in \mathrm{CL}(U\times\Theta)$ be arbitrary.

The functions above are ordinarily analytic, and hence their ambient extensions are given by the same defining formulas. In particular, the ambient extension Δ_e of Δ is such that, for every $h \in [1..n_\mathrm{h}]$,

$$(\Delta_\mathrm{e}(\overline{u};\overline{\theta}))_h := \mathrm{softplus}\left((\overline{W}_\Delta \cdot \overline{u} + \overline{b}_\Delta)_h\right) \;>\; 0. \tag{230}$$

Consequently, the ambient extension A_e of A is such that, for every $r \in [1..d_\mathrm{s}]$ and every $p \in [1..n_\mathrm{p}]$,

$$0 \;<\; (A_\mathrm{e}(\overline{u};\overline{\theta}))_{r,p} = \exp\left(-\exp(\overline{a})_{r,p}\cdot(\Delta_\mathrm{e}(\overline{u};\overline{\theta}))_p\right) \;<\; 1. \tag{231}$$

By Equation (228), we have that $A_\mathrm{e}(\overline{u};\overline{\theta})$ is diagonal, and therefore

$$\left\|A_\mathrm{e}(\overline{u};\overline{\theta})\right\| \;=\; \max_{\substack{r\in[1..d_\mathrm{s}]\\ p\in[1..n_\mathrm{h}]}} (A_\mathrm{e}(\overline{u};\overline{\theta}))_{r,p} \;<\; 1,$$

where the last inequality follows from Equation (231) and the fact that the maximum is taken over finitely many entries.

Since $(\overline{u},\overline{\theta})$ was arbitrary, we conclude that $\tilde{A}$ satisfies Equation (182) of Definition 273.

Since $i \in [1..\ell]$ and $j \in [1..m_i]$ were arbitrary, every Parametric Linear Recurrent Unit $\tilde{D}_{i,j}$ occurring in the flattened recurrent cascade presentation $\tilde{\mathsf{C}}$ satisfies both recurrent-unit requirements of Definition 273.

It remains to verify the required regularity of the layer output functions. Since C is a parametric recurrent neural cascade, we have that every connecting function g_i is analytic. Furthermore, since S is a parametric analytic output extension presentation, the parametric output function h is analytic.

By Definition 268, the first connecting function g_1 is unchanged; for every $i \in [2..\ell]$, the corresponding flattened connecting function satisfies

$$\tilde{g}_i\Big(\tilde{x}_\mathrm{pre}, u, \tilde{x}_\mathrm{cur};\; \theta_{g_i}\Big) \;=\; g_i\Big(\phi_{<i}^{-1}\big(\tilde{x}_\mathrm{pre}\big), u, \phi_{<i}^{-1}\big(\tilde{x}_\mathrm{cur}\big);\; \theta_{g_i}\Big);$$

and the flattened output function satisfies

$$\tilde{h}\Big(\tilde{x}_{\mathrm{pre}}, u, \tilde{x}_{\mathrm{cur}};\, \theta_h\Big) \;=\; h\Big(\phi_{\mathsf{C}}^{-1}\big(\tilde{x}_{\mathrm{pre}}\big), u, \phi_{\mathsf{C}}^{-1}\big(\tilde{x}_{\mathrm{cur}}\big);\, \theta_h\Big).$$

By Fact 59, every $\phi_{<i}^{-1}$ and ϕ_{C}^{-1} is linear and hence analytic. Therefore, by Proposition 1, every connecting function of $\tilde{\mathsf{C}}$ and the final output function $\tilde{h}$ are parametric analytic. By Fact 11, the corresponding layer output functions of $\tilde{S}$ are continuously-differentiable up to the closures of their domains.

Thus all conditions of Definition 273 hold for $\tilde{S}$. Therefore, $\tilde{S}$ is a smooth unitwise-contractive Linear RNN. □

T.3 Positive-margin expressivity of Mamba

Theorem 66 (Positive-margin expressivity of Mamba)**.** *Every language recognised with positive margin by a Mamba model is definite.* End of Theorem 66.

Proof. Let $S \in \mathcal{M}[\mathrm{Mamba}]$ recognise a language L with positive margin $\gamma \in \mathbb{R}^{>0}$.

By Definition 278, we have that S is a parametric $\mathcal{U}[\mathrm{Mamba}]$-RNC. Hence, by the definition of parametric RNCs, there exists a parametric recurrent cascade presentation C of type $\big(\mathcal{U}[\mathrm{Mamba}], \mathcal{F}_{\mathrm{AN}}^{\mathrm{P}}\big)$ such that

$$[\![\mathsf{C}]\!] = \mathrm{dyn}(S).$$

Let x^{init} be the initial state of S, and let h be its parametric output function. Since S is a parametric RNC, we have that h is analytic, and hence the output extension presentation

$$\mathsf{S} \;:=\; \langle \mathsf{C}, x^{\mathrm{init}} \rangle \blacktriangleright h$$

is an analytic output extension of S.

Let $\tilde{\mathsf{S}}$ be the Linear RNN presentation associated with S according to Definition 268, and let

$$\tilde{S} \;:=\; [\![\tilde{\mathsf{S}}]\!]$$

be the corresponding Linear RNN representation of S.

By Proposition 64, we have that S and $\tilde{S}$ have the same input-output behaviour under the same parameter values. Hence $\tilde{S}$ recognises L with the same margin γ.

By Theorem 65, we have that $\tilde{S}$ is a smooth unitwise-contractive Linear RNN.

Therefore, by Theorem 53, we conclude that L is definite. □

U Analysis of mLSTM

U.1 Definition of mLSTM

Definition 279 (Generic Stabilised Sigmoid-Forget mLSTM Recurrent Units)**.** Let $d_{\mathrm{k}}, d_{\mathrm{v}} \in \mathbb{N}^{>0}$. The *admissible stabilised sigmoid-forget mLSTM input space* of key-value dimensions $(d_{\mathrm{k}}, d_{\mathrm{v}})$ is

$$\mathcal{U}_{\mathrm{mLSTM}} := \mathbb{R}^{>0} \times (0,1) \times \mathbb{R}^{d_{\mathrm{k}}} \times \mathbb{R}^{d_{\mathrm{v}}}.$$

A *Generic Stabilised Sigmoid-Forget mLSTM Recurrent Unit* is dynamics

$$\hat{D} = \langle X, \hat{U}, \hat{f} \rangle, \qquad X \subseteq \mathbb{R}^{d_{\mathrm{v}} \times d_{\mathrm{k}}} \times \mathbb{R}^{d_{\mathrm{k}}} \times \mathbb{R}, \qquad \hat{U} \subseteq \mathcal{U}_{\mathrm{mLSTM}},$$

whose state is written as

$$x = \big(X_{\mathrm{c}}, x_{\mathrm{n}}, x_{\mathrm{m}}\big)$$

and whose input is written as

$$\hat{u} = \big(\bar{g}_{\mathrm{i}}, \bar{g}_{\mathrm{f}}, k, v\big).$$

For every $x \in X$ and $\hat{u} \in \hat{U}$, define

$$\begin{aligned} x'_{\mathrm{m}} &:= \max\left\{\log(\bar{g}_{\mathrm{f}}) + x_{\mathrm{m}}, \log(\bar{g}_{\mathrm{i}})\right\}, \\ g_{\mathrm{f}} &:= \exp\left(\log(\bar{g}_{\mathrm{f}}) + x_{\mathrm{m}} - x'_{\mathrm{m}}\right), \\ g_{\mathrm{i}} &:= \exp\left(\log(\bar{g}_{\mathrm{i}}) - x'_{\mathrm{m}}\right), \end{aligned}$$

and

$$\begin{aligned} X'_{\mathrm{c}} &:= g_{\mathrm{f}} \cdot X_{\mathrm{c}} + g_{\mathrm{i}} \cdot v \cdot k^{\top}, \\ x'_{\mathrm{n}} &:= g_{\mathrm{f}} \cdot x_{\mathrm{n}} + g_{\mathrm{i}} \cdot k. \end{aligned}$$

Then

$$\hat{f}\left(\big(X_{\mathrm{c}}, x_{\mathrm{n}}, x_{\mathrm{m}}\big), \big(\bar{g}_{\mathrm{i}}, \bar{g}_{\mathrm{f}}, k, v\big)\right) := \big(X'_{\mathrm{c}}, x'_{\mathrm{n}}, x'_{\mathrm{m}}\big).$$

We call X_{c}, x_{n}, and x_{m} respectively the *memory state*, *normaliser state*, and *stabiliser state*.

End of Definition 279.

Definition 280 (Stabilised Sigmoid-Forget mLSTM)**.** Let $d_{\mathrm{in}}, d_{\mathrm{k}}, d_{\mathrm{v}} \in \mathbb{N}^{>0}$, and let

$$\hat{D} = \langle X, \hat{U}, \hat{f} \rangle$$

be a Generic Stabilised Sigmoid-Forget mLSTM Recurrent Unit with key-value dimensions $(d_{\mathrm{k}}, d_{\mathrm{v}})$.

A *Stabilised Sigmoid-Forget mLSTM Recurrent Unit* is parametric dynamics

$$D = \langle X, U, f; \Theta \rangle, \qquad U \subseteq \mathbb{R}^{d_{\mathrm{in}}},$$

having components as described next.

▸ **SubDefinition 280.1** (Parameter space).

The parameter space is a finite-dimensional Euclidean space

$$\Theta = \Theta_{\mathrm{k}} \times \Theta_{\mathrm{v}} \times \Theta_{\mathrm{i}} \times \Theta_{\mathrm{f}}.$$

End of SD.280.1 ◂

▸ **SubDefinition 280.2** (Recurrence-input functions).

The recurrence-input functions are parametric feedforward neural maps

$$\begin{aligned} k &: U \times \Theta_{\mathrm{k}} \to \mathbb{R}^{d_{\mathrm{k}}}, & v &: U \times \Theta_{\mathrm{v}} \to \mathbb{R}^{d_{\mathrm{v}}}, \\ \widetilde{g}_{\mathrm{i}} &: U \times \Theta_{\mathrm{i}} \to \mathbb{R}, & \widetilde{g}_{\mathrm{f}} &: U \times \Theta_{\mathrm{f}} \to \mathbb{R}, \end{aligned}$$

with raw gate functions

$$\bar{g}_{\mathrm{i}} : U \times \Theta_{\mathrm{i}} \to \mathbb{R}^{>0}, \qquad \bar{g}_{\mathrm{i}}(u;\theta_{\mathrm{i}}) := \exp\left(\widetilde{g}_{\mathrm{i}}(u;\theta_{\mathrm{i}})\right),$$
$$\bar{g}_{\mathrm{f}} : U \times \Theta_{\mathrm{f}} \to (0,1), \qquad \bar{g}_{\mathrm{f}}(u;\theta_{\mathrm{f}}) := \sigma\left(\widetilde{g}_{\mathrm{f}}(u;\theta_{\mathrm{f}})\right).$$

End of SD.280.2 ◂

▸ **SubDefinition 280.3** (Dynamics function).

For

$$\theta = \left(\theta_{\mathrm{k}}, \theta_{\mathrm{v}}, \theta_{\mathrm{i}}, \theta_{\mathrm{f}}\right) \in \Theta,$$

the parametric dynamics function is obtained by applying the dynamics function of $\hat{D}$ to the parametrised recurrence input:

$$f(x,u;\theta) := \hat{f}\Big(x,\ \left(\bar{g}_{\mathrm{i}}(u;\theta_{\mathrm{i}}), \bar{g}_{\mathrm{f}}(u;\theta_{\mathrm{f}}), k(u;\theta_{\mathrm{k}}), v(u;\theta_{\mathrm{v}})\right)\Big).$$

End of SD.280.3 ◂

▸ **SubDefinition 280.4** (Models and classes).

An *mLSTM*$(\exp,\sigma;1,1)$ *model* is a parametric Recurrent Neural Cascade whose recurrent units are Stabilised Sigmoid-Forget mLSTM Recurrent Units and whose recurrent initial states are zero.

We denote the class of such parametric systems by

$$\mathfrak{S}_{\mathrm{mLSTM}(\exp,\sigma;1,1)}.$$

End of SD.280.4 ◂

End of Definition 280.

U.2 Positive-Margin Expressivity

The complete stabilised mLSTM recurrence is not strictly contractive: when the first branch of the stabiliser maximum is active, the stabiliser state is propagated with coefficient one. Nevertheless, sigmoid forget gating yields a stronger property sufficient for the present expressivity analysis: dependence on an arbitrarily remote prefix vanishes uniformly. We establish this first for a single recurrent unit.

Lemma 48 (Fading memory of stabilised sigmoid-forget mLSTM units)**.** *Let D be an instance of a Stabilised Sigmoid-Forget mLSTM Recurrent Unit, and let K be a bounded subset of its input space.*

Consider an arbitrary family $\mathcal{P}$ of pairs of input sequences

$$(u,u') \in K^{\omega} \times K^{\omega},$$

together with initial-state pairs (x_0, x_0') ranging over a bounded set. Suppose that

$$\eta_t := \sup_{(u,u')\in\mathcal{P}} \|u_t - u_t'\| \to 0 \qquad \text{as } t \to \infty. \tag{232}$$

Then the corresponding state trajectories satisfy

$$\sup_{(u,u')\in\mathcal{P}} \|x_t - x_t'\| \to 0 \qquad \text{as } t \to \infty, \tag{233}$$

uniformly over the bounded set of initial-state pairs. End of Lemma 48.

Proof. Fix the parameter value of the considered instance and define

$$a(u) := \log\left(\bar{g}_{\mathrm{f}}(u)\right), \qquad b(u) := \log\left(\bar{g}_{\mathrm{i}}(u)\right).$$

Since

$$\bar{g}_{\mathrm{f}}(u) = \sigma\left(\widetilde{g}_{\mathrm{f}}(u)\right),$$

and the recurrence-input functions extend continuously to the compact closure of K, there exist constants

$$\delta \in \mathbb{R}^{>0}, \qquad b_-, b_+ \in \mathbb{R}$$

such that

$$a(u) \leq -\delta, \qquad b_- \leq b(u) \leq b_+ \qquad \forall u \in K. \tag{234}$$

Consider first the stabiliser state. Its recurrence is

$$x_{\mathrm{m},t} = \max\left\{a(u_t) + x_{\mathrm{m},t-1}, b(u_t)\right\}. \tag{235}$$

Expanding Equation (235) gives

$$x_{\mathrm{m},t} = \max\left\{ x_{\mathrm{m},0} + \sum_{j=1}^{t} a(u_j), \right. \\ \left. b(u_k) + \sum_{j=k+1}^{t} a(u_j) \;\middle|\; k \in [1..t] \right\}. \tag{236}$$

Let M_{m} uniformly bound the initial stabiliser states under consideration. Choose $N_{\mathrm{m}} \in \mathbb{N}^{>0}$ sufficiently large that

$$M_{\mathrm{m}} - N_{\mathrm{m}}\delta < b_-, \qquad b_+ - N_{\mathrm{m}}\delta < b_-. \tag{237}$$

By Equations (234) and (237), for every $t \geq N_{\mathrm{m}}$, neither the candidate originating from $x_{\mathrm{m},0}$ nor a candidate introduced more than N_{m} steps earlier can attain the maximum in Equation (236). Therefore

$$x_{\mathrm{m},t} = \max_{k \in [t-N_{\mathrm{m}}+1..t]} \left\{ b(u_k) + \sum_{j=k+1}^{t} a(u_j) \right\}. \tag{238}$$

Thus the stabiliser is not contractive in one step, but after N_{m} steps it is independent of its preceding state and depends only on a uniformly bounded window of recurrent inputs.

Since a and b are uniformly continuous on the compact closure of K, Equation (232) and the finite-window representation Equation (238) imply

$$|x_{\mathrm{m},t} - x'_{\mathrm{m},t}| \to 0 \tag{239}$$

uniformly over the considered family.

We next consider the memory and normaliser states. Define their rescaled forms by

$$\bar{X}_{\mathrm{c},t} := \exp(x_{\mathrm{m},t}) \cdot X_{\mathrm{c},t}, \qquad \bar{x}_{\mathrm{n},t} := \exp(x_{\mathrm{m},t}) \cdot x_{\mathrm{n},t}.$$

Direct substitution in the dynamics equations gives

$$\bar{X}_{\mathrm{c},t} = \bar{g}_{\mathrm{f}}(u_t) \cdot \bar{X}_{\mathrm{c},t-1} + \bar{g}_{\mathrm{i}}(u_t) \cdot v(u_t) \cdot k(u_t)^{\top}, \tag{240}$$

$$\bar{x}_{\mathrm{n},t} = \bar{g}_{\mathrm{f}}(u_t) \cdot \bar{x}_{\mathrm{n},t-1} + \bar{g}_{\mathrm{i}}(u_t) \cdot k(u_t). \tag{241}$$

Compactness and sigmoid forget gating give a constant

$$\alpha \in [0, 1) \qquad \text{such that} \qquad \bar{g}_{\mathrm{f}}(u) \leq \alpha \qquad \forall u \in K. \tag{242}$$

Moreover, all additive terms in Equations (240) and (241) are bounded and uniformly continuous on the compact closure of K. Hence the rescaled state trajectories are uniformly bounded.

Stack the rescaled memory and normaliser states into a vector $\bar{x}_t$. Comparing two trajectories gives an estimate

$$\|\bar{x}_t - \bar{x}'_t\| \leq \alpha \|\bar{x}_{t-1} - \bar{x}'_{t-1}\| + e_t, \tag{243}$$

where

$$e_t \to 0$$

uniformly over the considered family, by Equation (232), uniform boundedness of the rescaled states, and uniform continuity of the recurrence-input functions.

Iterating Equation (243) gives

$$\|\bar{x}_t - \bar{x}'_t\| \leq \alpha^t \|\bar{x}_0 - \bar{x}'_0\| + \sum_{s=1}^{t} \alpha^{t-s} e_s.$$

Since $\alpha < 1$ and $e_s \to 0$ uniformly, both terms converge uniformly to zero. Therefore

$$\|\bar{x}_t - \bar{x}'_t\| \to 0. \tag{244}$$

Finally, Equation (236) and Equation (234) give uniform upper and lower bounds on the stabiliser states. Consequently, the inverse rescaling

$$\big(\bar{X}_{\mathrm{c}}, \bar{x}_{\mathrm{n}}, x_{\mathrm{m}}\big) \longmapsto \big(\exp(-x_{\mathrm{m}})\bar{X}_{\mathrm{c}}, \exp(-x_{\mathrm{m}})\bar{x}_{\mathrm{n}}, x_{\mathrm{m}}\big)$$

is uniformly continuous on the bounded set under consideration. Equations (239) and (244) therefore imply Equation (233). □

Lemma 49 (Uniform suffix forgetting of stabilised sigmoid-forget mLSTM RNCs)**.** *Let S be an instance of an mLSTM$(\exp, \sigma; 1, 1)$ model, and let*

$$\mathrm{enc} : \Sigma \to U$$

be an encoder from a finite alphabet Σ.

Then, for some sequence

$$(\varepsilon_n)_{n \in \mathbb{N}^{>0}} \in \mathbb{R}^{\geq 0 \omega} \qquad \textit{with} \qquad \varepsilon_n \to 0 \quad \textit{as} \quad n \to \infty,$$

we have

$$d_Y\Big(S\big(\mathrm{enc}(vz)\big), S\big(\mathrm{enc}(wz)\big)\Big) \leq \varepsilon_n \tag{245}$$

for every

$$n \in \mathbb{N}^{>0}, \qquad v, w \in \Sigma^*, \qquad z \in \Sigma^n.$$

End of Lemma 49.

Proof. Since Σ is finite, the encoded input set $\mathrm{enc}(\Sigma)$ is bounded.

Consider the first recurrent layer. Lemma 48, applied with zero initial states and bounded inputs, first gives uniform boundedness of its recurrent states over all encoded words. The corresponding feedforward layer-output maps therefore have bounded reachable argument sets and bounded outputs.

Consider now two computations which have processed arbitrary prefixes $v, w \in \Sigma^*$ and subsequently process the same suffix

$$z = z_1 \cdots z_n.$$

During this common suffix, every recurrent unit in the first layer receives exactly the same input in the two computations. Thus Equation (232) holds with $\eta_t = 0$. Lemma 48 consequently implies that the first-layer recurrent states converge uniformly to one another as the length of the common suffix tends to infinity. Uniform continuity of the corresponding layer-output map on the bounded reachable argument set yields the same conclusion for the first-layer outputs.

Proceed inductively through the finite cascade. Suppose that the outputs of one layer in the two computations converge uniformly to one another during increasingly long common suffixes. These outputs determine the recurrent inputs u_t and u'_t received by the next layer. Uniform continuity of the corresponding inter-layer map therefore gives

$$\|u_t - u'_t\| \to 0$$

uniformly over prefixes and common suffixes. Lemma 48 then implies uniform convergence of the recurrent states of the next layer, and uniform continuity of its output map implies uniform convergence of its outputs.

Since the cascade contains finitely many layers, the final outputs satisfy

$$\sup_{\substack{v,w \in \Sigma^* \\ z \in \Sigma^n}} d_Y\Big(S\big(\mathrm{enc}(vz)\big),\, S\big(\mathrm{enc}(wz)\big)\Big) \to 0 \qquad \text{as } n \to \infty.$$

Defining the displayed supremum to be ε_n proves the claim. □

Theorem 67 (Positive-margin expressivity of stabilised sigmoid-forget mLSTM RNCs)**.** *The positive-margin formal-language expressivity of mLSTM*$(\exp, \sigma; 1, 1)$ *RNCs is contained in the definite languages:*

$$\mathrm{Expr}_{+\mathrm{margin}}\left(\mathfrak{S}_{\mathrm{mLSTM}(\exp,\sigma;1,1)}\right) \subseteq \mathsf{Def}. \tag{246}$$

End of Theorem 67.

Proof. Let S be an instance of an mLSTM$(\exp, \sigma; 1, 1)$ model that recognises a language L over a finite alphabet Σ with positive margin, through an encoder enc and decoder dec. Let $\gamma \in \mathbb{R}^{>0}$ be a corresponding decoder margin.

By Lemma 49, choose $N \in \mathbb{N}^{>0}$ such that

$$\varepsilon_N < \gamma.$$

Then, for every $v, w \in \Sigma^*$ and $z \in \Sigma^N$,

$$d_Y\Big(S\big(\mathrm{enc}(vz)\big),\, S\big(\mathrm{enc}(wz)\big)\Big) < \gamma.$$

By the decoder-margin condition,

$$\mathrm{dec}\Big(S\big(\mathrm{enc}(vz)\big)\Big) = \mathrm{dec}\Big(S\big(\mathrm{enc}(wz)\big)\Big),$$

and hence

$$vz \in L \quad \Longleftrightarrow \quad wz \in L.$$

Lemma 4 therefore implies that L is definite. Since S and L were arbitrary, Equation (246) follows. □

Remark 54 (The result does not rely on strict contractivity)*.* Stabilised sigmoid-forget mLSTM dynamics are not strictly contractive in their complete state: while the first branch of the stabiliser maximum is active, the stabiliser propagates its preceding value with coefficient one.

The preceding result instead follows from two complementary mechanisms. The stabiliser has a uniformly bounded memory horizon because every continuation of an old candidate accumulates

$$\log(\bar{g}_{\mathrm{f}}) \leq -\delta < 0,$$

whereas the rescaled memory and normaliser states are uniformly contractive because

$$\bar{g}_{\mathrm{f}} \leq \alpha < 1.$$

Together these properties yield uniform suffix forgetting and hence the definite-language upper bound under positive-margin recognition. *End of Remark 54.*

Remark 55 (Relation to the official mLSTM implementation)*.* The stabilised sigmoid-forget mLSTM considered above is in line with the recurrent mechanism used by the current official NX-AI xLSTM implementation [70]. In particular, its recurrent backend applies a sigmoid forget gate in log-space, updates the stabiliser according to

$$x'_{\mathrm{m}} = \max\left\{\log(\bar{g}_{\mathrm{f}}) + x_{\mathrm{m}}, \widetilde{g}_{\mathrm{i}}\right\},$$

and uses the corresponding stabilised gates

$$g_{\mathrm{f}} = \exp\left(\log(\bar{g}_{\mathrm{f}}) + x_{\mathrm{m}} - x'_{\mathrm{m}}\right), \qquad g_{\mathrm{i}} = \exp\left(\widetilde{g}_{\mathrm{i}} - x'_{\mathrm{m}}\right).$$

Thus, in the notation of Definition 280, the implemented recurrent mechanism corresponds to the $\mathrm{mLSTM}(\exp, \sigma; 1, 1)$ variant considered here.

The official implementation additionally includes architectural details such as multi-head organisation, query-dependent readout, normalisation, and optimised parallel or chunkwise kernels. These do not change the input-gate, forget-gate, and stabiliser recurrence underlying the preceding positive-margin expressivity result. *End of Remark 55.*

Computational Resources

We estimate to have used about 80 days of computation time on a single A100 GPU.

NeurIPS Paper Checklist

(1) **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract is a faithful account of the contents of the paper.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

(2) **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We include a dedicated paragraph describing limitations.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

(3) **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All assumptions are clearly stated. Full proofs of all results are given in the appendix, with proof sketches of all results in the main body.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

(4) **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Details in paper, dedicated appendices, and in the code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (1) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (2) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (3) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (4) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

(5) **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code is provided, but not with an open source licence.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

(6) **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide details of the experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

(7) **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The reported accuracies are means of 99% confidence intervals with specified maximum margin of error. In the appendix we specify how they CIs are obtained.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

(8) **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We specify the computational resources in the experiments' section, and we provide an estimate of the total computation time.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

(9) **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research is in line with the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

(10) **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The research is at an early stage, at which point there is no clear societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

(11) **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

(12) **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The only assets we use are (i) the Mamba and the models of the xLSTM family, both released under The Apache License 2.0, that allows users to freely use, modify, and distribute software for any purpose, including commercial projects, and (ii) and OpenWebText that is released under CC0, allowing it to be used, modified, and distributed for any purpose, including commercial projects, without restriction.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

(13) **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A].

Justification: the paper does not release new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

(14) **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Not relevant for this paper.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

(15) **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A].

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

(16) **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: LLMs are not a components of the contribution.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.