Hard Attention Transformers and BSS-Machines

Lena Ehrmuth
Institute for Theoretical Computer Science
Leibniz University Hanover
Hanover, Germany
ehrmuth@thi.uni-hannover.de

Abstract

We define a new variant of transformers called recursive transformers, which generate sequences of vectors without discretisation in each step. We show that over ordered ring extensions of the integers recursive transformers with circuit activation functions and hard attention are computationally equivalent to the well-established algebraic model of BSS-machines. If the transformers use feedforward networks instead, they are equivalent to linear BSS-machines with degree-2 branching.

1 Introduction

The relation between recursive neural networks and Turing machines was first described in the work by Siegelmann and Sontag [1995]. They showed that a single layer network with sigmoid activation can be used to simulate a Turing machine. Given this result, it is not surprising that transformers, the neural network architecture defined by Vaswani et al. [2017], can also be used to simulate Turing machines. Pérez et al. [2021] showed this for hard attention transformers using absolute values of dot-products as attention functions. Merrill and Sabharwal [2024] had the same result for soft attention transformers with a projected pre-norm and strict future masking.

Recent analyses of transformer expressivity, like Hao et al. [2022], Strobl [2023] and Merrill and Sabharwal [2023], have mostly been focused on comparisons to Boolean circuit classes. Another approach involves logical characterizations, for example Chiang et al. [2023], Barceló et al. [2024] and Yang et al. [2024]. A survey of recent results has been published by Strobl et al. [2024].

Motivation All of the previously mentioned results use Boolean functions or languages over a finite alphabet for comparison. But, internally the transformer model can be defined over different algebraic structures, like the real numbers, which gives rise to the question, whether they can even simulate algebraic models of computation. Several such models have been defined. The definition of circuits can be extended from the Boolean semiring to any other semiring, so they use addition and multiplication gates instead conjunction and disjunction. These kinds of circuits are called arithmetic and they compute polynomial functions, but they can also be defined with additional gate types for division or sign. Barlag et al. [2024] have recently analysed their relation to graph neural networks and this was also done in a similar way for transformers by Ehrmuth and Strieker [2026].

In this paper, we analyse the relation between transformers and BSS-machines, a model introduced by Blum, Shub, and Smale [1989]. A BSS-machine has an infinite tape with cells of numbers over some ordered semiring and a finite set of states. In each state, it can evaluate a polynomial function on a fixed number of tape cells, shift the tape one cell to the left or right or branch to one of two successor states depending on whether the value of a polynomial is ≥ 0. When the underlying semiring is the Boolean semiring, the integers or the rationals, this model is Turing-complete, but when using the real numbers, it becomes strictly more powerful. For an overview of complexity results using this model, see Blum et al. [1998]. We believe that this model, which has been subject to theoretical research for over 30 years, is well suited for comparison with the transformer architecture, because it exactly

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

describes arithmetic operations even when using real numbers. Of course, any model using reals can never be simulated by an actual computer, but our results also hold over other rings like the rationals.

To simulate a BSS-machine, the transformers must be able to output numbers directly instead of a finite set of tokens. We define a variant of transformers called recursive transformers, which take a sequence of vectors over some ring as an input and also output a vector sequence. They are applied recursively until a specific component is non-zero, increasing the length of the sequence by one at each step. For the attention scheme we use leftmost hard attention, where each vector attends to exactly one vector with a maximal attention score. The rightmost variant would also work, but using soft attention would make it difficult to transfer exact real values between vectors.

Main Results The main contribution of this paper is an equivalence between the functions computed by recursive transformers using hard attention and BSS-machines. However, there are two variants of this result. To simulate any BSS-machine, we need a transformer that can compute polynomial functions, or arithmetic circuits. We allow for the use arithmetic circuits instead of feedforward networks. These circuit transformers are equivalent to standard BSS-machines, but if we restrict the transformers to the usual feedforward networks, we obtain an equivalence to linear BSS-machines with degree-2 branching.

Because this equivalence holds both over $\mathbb{Q}$ and $\mathbb{R}$, we immediately get a difference in expressivity between hard attention transformers with and without exact real arithmetic. BSS-machines over $\mathbb{Q}$ are Turing-complete while machines over $\mathbb{R}$ can compute some functions that are not Turing-computable. The construction used to simulate BSS-machines is the same for both fields. The only difference is allowing irrational constants and input values. Therefore, the difference in computational power emerges precisely from the exact representation and arithmetic for irrational numbers.

Organization In the next section, we introduce the models of transformers and BSS-machines. In Section 3, we briefly explain how transformers can be simulated by BSS-machines. Section 4 contains a detailed construction to simulate a BSS-machines with circuit transformers and Section 5 extpands on that to make it compatible with feedforward network activation functions. Full proofs for all statements can be found in the appendix.

2 Preliminaries

We define $\mathbb{N} = \{0, 1, 2, \dots\}$. For a set S, we use S^* for the Kleene closure of S. We use lowercase letters to denote numbers, overlines for vectors and uppercase letters for sequences of vectors. The length of a vector or sequence is denoted with $|\overline{x}|$ or $|X|$.

2.1 Algebra

Transformers and BSS-machines can be defined over any semiring, but in this paper, we are only concerned with rings.

Definition 2.1. A *ring* is a set R equipped with binary functions $+$ and $\times$ such that $(R, +)$ is an abelian group with identity 0, $(R, \times)$ is a monoid with identity 1, 0 is absorbing with respect to $\times$ and $\times$ distributes over $+$.

We will need an order on the values to define hard attention and if we want to simulate the computation of one ring within another, we need what we call a ring extension.

Definition 2.2. An *ordered* ring is a ring R with a partial order $\leq \subseteq R^2$ such that for all $x, y, z \in R$ we get $x \leq y \;\Rightarrow\; x + z \leq y + z$ and $x \leq y \wedge 0 \leq z \;\Rightarrow\; x \times z \leq y \times z \wedge z \times x \leq z \times y$.

Definition 2.3. Let R and T be rings with functions $+_R, \times_R$ and $+_T, \times_T$ having identities $0_R, 1_R$ and $0_T, 1_T$. Then a function $f\colon R \to T$ is a *homomorphism* if we get

$$f(x +_R y) = f(x) +_T f(y), \quad f(x \times_R y) = f(x) \times_T f(y), \quad \text{and} \quad f(1_R) = 1_T$$

for all $x, y \in R$. We call T a *ring extension* of R if there is an injective homomorphism from R to T. We call a ring extension of $\mathbb{Z}$ *$\mathbb{Z}$-ordered*, if it preserves the usual order of $\mathbb{Z}$, which means that for all $x, y \in \mathbb{Z}$, $f(x) \leq f(y) \Leftrightarrow x \leq y$. For example, $\mathbb{Q}$ and $\mathbb{R}$ are $\mathbb{Z}$-ordered rings.

2.2 Transformers

We adapt a definition by Hao et al. [2022] to describe transformer encoders over arbitrary rings.

Definition 2.4. A *generalized transformer* with K layers and H attention heads is a tuple $(R, d, \Sigma, f^{\text{in}}, f^{\text{pos}}, f^{\text{att}}_{1,1}, f^{\text{att}}_{1,2}, \ldots, f^{\text{att}}_{K,H}, f^{\text{pool}}_{1,1}, \ldots, f^{\text{pool}}_{K,H}, f^{\text{act}}_{1}, \ldots, f^{\text{act}}_{K})$ where R is the underlying ring, $d \in \mathbb{N}$ is the dimension, Σ is the (possibly infinite) input alphabet, $f^{\text{in}}\colon \Sigma \to R^d$ is the input embedding, $f^{\text{pos}}\colon \mathbb{N} \times \mathbb{N} \to R^d$ is the positional embedding, $f^{\text{att}}_{k,h}\colon R^d \times R^d \to R$ is the attention function for attention head h in layer k, $f^{\text{pool}}_{k,h}\colon (R^d)^* \times R^* \to R^d$ is the pooling function for attention head h in layer k and $f^{\text{act}}_{k}\colon (R^d)^{H+1} \to R^d$ is the activation function for layer k. Such a transformer computes a function $T\colon \Sigma^* \to (R^d)^*$ defined by induction over k. First, we add an extra vector to the input sequence $X = (x_1, \ldots, x_n) \in \Sigma^*$ and turn it into a sequence of R^d-vectors using the input and positional embedding.

$$Y^0 = \big(f^{\text{in}}(x_1) + f^{\text{pos}}(1, n+1), \ldots, f^{\text{in}}(x_n) + f^{\text{pos}}(n, n+1), f^{\text{in}}(0) + f^{\text{pos}}(n+1, n+1)\big)$$

Each layer k transforms the previous sequence Y^{k-1} into a sequence Y^k of the same length in the following way. The attention function of each attention head h produces a matrix of attention values.

$$a^{k,h}_{i,j} = f^{\text{att}}_{k,h}(Y^{k-1}_i, Y^{k-1}_j) \quad \text{for } 1 \le i, j \le n+1$$

The pooling function of each attention head h turnes it back into a vector sequence.

$$Z^{k,h}_i = f^{\text{pool}}_{k,h}\big(Y^{k-1}, (a^{k,h}_{i,1}, \ldots, a^{k,h}_{i,n+1})\big) \quad \text{for } 1 \le i \le n+1$$

Finally, the activation function of the layer is applied to each vector in those sequences.

$$Y^k_i = f^{\text{act}}_k\big(Y^{k-1}_i, Z^{k,1}_i, \ldots, Z^{k,H}_i\big) \quad \text{for } 1 \le i \le n+1$$

The output of the transformer is simply the output of the last layer, $T(X) = Y^K$.

Definition 2.5. Let R be a ring and A, B, C, D, E be sets of functions. Then $\mathbf{FT}_R\big(A, B, C, D, E\big)$ is the set of functions computed by transformers using R as the underlying ring, input embeddings from A, positional embeddings from B, activation functions from C, attention functions from D and pooling functions from E.

The definition of dot-product attention by Vaswani et al. [2017] is described by the following sets.

Definition 2.6. Let R be a ring and $f\colon R^* \to R^*$. We define the set of dot-product attention functions over R and the set of pooling functions using weighted sums over R.

$$\mathbf{DPA}_R = \left\{ \begin{array}{l} g\colon R^d \times R^d \to R, \\ (\overline{x}, \overline{y}) \mapsto A\overline{x} \cdot B\overline{y} \end{array} \middle| \begin{array}{c} d \in \mathbb{N} \text{ and} \\ A, B \in R^{d \times d} \end{array} \right\} \mathbf{WS}_R(f) = \left\{ \begin{array}{l} g\colon (R^d)^* \times R^* \to R^d, \\ (X, \overline{a}) \mapsto \sum_{i=1}^{|X|} f(\overline{a})_i \times X_i \end{array} \middle| \, d \in \mathbb{N} \right\}$$

We also use a simple integer positional embedding, which sets one vector coordinate x_p to $f^{\text{pos}}_R(i, n)_p = \sum_1^i 1_R$ and all other coordinates to zero. Then $\mathbf{IPE}_R = \{f^{\text{pos}}_R\}$.

One broadly used variant of activation functions is given by feedforward-networks.

Definition 2.7. Let R be an ordered ring. Then $\mathbf{FF}_R$ is the set of functions $f\colon R^m \to R^n$ computed by feedforward networks using the ReLU activation function and any number of layers.

We use a second kind of activation function which is defined by arithmetic circuits. These circuits use addition and multiplication gates to compute polynomial functions over a ring. We also use the function

$$\operatorname{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

in some cases. For a broader introduction to this topic, see Vollmer [1999].

Definition 2.8. Let R be a ring. Then $\mathbf{FC}_{\mathbf{IRC}}{}_R$ is the set of functions $f\colon R^m \to R^n$ computed by circuits over the basis $(R, \{+, \times\})$. If R is ordered, we also define $\mathbf{FC}_{\mathbf{IRC}}{}_R[\operatorname{sign}]$ using circuits over $(R, \{+, \times, \operatorname{sign}\})$. Our definition of circuits also allows gates for arbitrary constants from R.

2.3 BSS-Machines

BSS-machines are a model of computation developed by Blum, Shub, and Smale, which behaves similarly to Turing machines. They define a finite number of states and work with an infinitely expanding memory or tape, but instead of symbols from a finite alphabet it contains elements of an underlying (semi-)ring. In a single step, a BSS-machine can evaluate a predefined polynomial function and branching of states is based on comparisons of polynomials.

Definition 2.9. Let R be an ordered ring. A **BSS**$_R$*-machine* is defined by a labeled directed graph of states with the following properties.

- There is exactly one input state z_{in} with no incoming and one outgoing edge.
- There is exactly one output state z_{out} with one incoming and no outgoing edge.
- Computation states have an associated polynomial function $p\colon R^m \to R^n$ and one outgoing edge.
- Shift states can be either left or right and have one outgoing edge.
- Branching states states have an associated polynomial $p\colon R^m \to R$, one outgoing edge labeled 0 and one outgoing edge labeled 1.

Definition 2.10. The tape contents of a **BSS**$_R$-machine are an element of R_∞, the bi-infinite direct sum space over R, i.e. $x = (\dots, x_{-2}, x_{-1}, x_0.\, x_1, x_2, \dots)$, where $x_i \in R$ for all $i \in \mathbb{Z}$. At any given point during the computation, only finitely many coordinates can be non-zero.

The tape cells are initially all zero. At the start of a computation, an input of length n is written to the tape cells $x_1, \dots, x_n$ and to distinguish it from the zeros, the cells $x_{1-n}, \dots, x_0$ are set to one. We call x_1 the head position of the tape. A complete configuration of a **BSS**$_R$-machine is described by its state z and the tape contents $x \in R_\infty$ and its computation is defined by the following relation.

Definition 2.11. For a non-banching state z, let $s(z)$ be the unique successor in the graph.

- For the input state, $(z_{in}, x) \vdash (s(z_{in}), x)$.
- If z is a left shift state, $(z, (\dots, x_0.\, x_1, \dots)) \vdash (s(z), (\dots, x_1.x_2, \dots))$.
- If z is a right shift state, $(z, (\dots, x_0.\, x_1, \dots)) \vdash (s(z), (\dots, x_{-1}.x_0, \dots))$.
- If z is a computation state with function $p\colon R^m \to R^n$, $(z, (\dots, x_0.\, x_1, \dots)) \vdash (s(z), (\dots, x_0.\, y_1, \dots, y_n, x_{n+1}, \dots))$, where $\overline{y} = p(x_1, \dots, x_m)$.
- If z is a branching state with polynomial $p\colon R^m \to R$, let z_0, z_1 be the unique successors through edges labeled $0, 1$. Then $(z, (\dots, x_0.\, x_1, \dots)) \vdash (z_1, (\dots, x_0.\, x_1, \dots))$ if $p(x_1, \dots, x_m) \geq 0$ and $(z, (\dots, x_0.\, x_1, \dots)) \vdash (z_0, (\dots, x_0.\, x_1, \dots))$, otherwise.

We define $a \vdash^* b$, if and only if there is a finite sequence, such that $a = s_0 \vdash s_1 \vdash \dots \vdash s_t = b$.

Definition 2.12. The input map $\mathcal{I}_\infty\colon R^* \to R_\infty$ is defined as

$$\mathcal{I}_\infty(\overline{x}) = (\dots, 0, \underbrace{1, \dots, 1}_{|\overline{x}|}.\, x_1, \dots, x_{|\overline{x}|}, 0, \dots)$$

and the output map $\mathcal{O}_\infty\colon R_\infty \to R^*$ is defined as

$$\mathcal{O}_\infty((\dots, x_{-1}, x_0.\, x_1, x_2, \dots)) = (x_1, \dots, x_n) \text{ where } n = \min\{i \mid i \geq 0, x_{-i} = 0\}.$$

Definition 2.13. The *halting set* of a **BSS**$_R$-machine is

$$H = \{X \mid X \in R^* \text{ and there is a } y \in R_\infty \text{ such that } (z_{in}, \mathcal{I}_\infty(X)) \vdash^* (z_{out}, y)\}.$$

A **BSS**$_R$-machine *computes* a partial function $f\colon R^* \to R^*$ defined on H as $f(X) = \mathcal{O}_\infty(y)$.

Definition 2.14. In a *linear* **BSS**$_R$-machine, all computation states have polynomials of degree ≤ 1. Such a machine uses *degree k branching*, if all branching states have polynomials of degree $\leq k$.

2.4 Recursion

Because transformers use a constant number of dimensions, they can only store a linear number of values with respect to the input length. To compute functions that require more space, we use transformers recursively, meaning the output is fed back into the transformer until a halting condition is met. This allows for the generation of sequences of arbitrary length.

Definition 2.15. Let R be a ring and $f\colon (R^d)^* \to (R^d)^*$. Using the halting function $h\colon (R^d)^* \to \{0,1\}$ with $h(\overline{x}^1, \dots, \overline{x}^n) = 1 \Leftrightarrow x^n_{hlt} \neq 0$ we define a partial function $\mathbf{R}(f)\colon (R^d)^* \to (R^d)^*$ by recursion:

$$\mathbf{R}(f)(X) = f^t(X) \quad \text{if } t = \min\{s \mid h(f^s(X)) = 1, s \in \mathbb{N}\} \text{ exists}$$

In natural language processing, transformers use a different kind of recursion, where only the last element in the output of the transformer is used and appended to the previous sequence. We will later show that this does not make a difference in expressivity.

Definition 2.16. Let R be a ring and $f\colon (R^d)^* \to (R^d)^*$. Using the same halting function h we define a partial function $\mathbf{S}(f)\colon (R^d)^* \to (R^d)^*$ by recursion:

$$\begin{aligned} Y^0 &= X \\ Y^{n+1} &= Y^n \mid (f(Y^n)_{|f(Y^n)|}) && \text{where } \mid \text{ denotes concatenation of sequences} \\ \mathbf{S}(f)(X) &= Y^t && \text{if } t = \min\{s \mid h(Y^s) \neq 0\} \text{ exists} \end{aligned}$$

As these recursive functions can only increase the length of a sequence, we need a way to remove parts of it. To allow the transformers to indicate, which vectors are supposed to be part of the output, we designate a coordinate x_{trim} as an indicator. Let $\mathrm{trim}\colon (R^d)^* \to (R^d)^*$ be the function that removes all vectors with $x_{\mathrm{trim}} = 0$ from the sequence without changing the order.

Definition 2.17. Let R be a ring, $\mathfrak{C}$ be a class of functions $(R^d)^* \to (R^d)^*$ and $f\colon R^* \to R^*$ a partial function. Then $f \in$ **R-$\mathfrak{C}$** if there is an input projection $p_{in}\colon R \to R^d$, a function $g \in \mathfrak{C}$ and an output projection $p_{out}\colon R^d \to R$ such that

$$f = p^*_{out} \circ \mathrm{trim} \circ \mathbf{R}(g) \circ p^*_{in}.$$

Here p^* denotes the element-wise application of p. The class **S-$\mathfrak{C}$** is defined analogously for $\mathbf{S}(g)$.

3 Simulating Transformers with BSS-machines

The definition of the transformer architecture can easily be translated into a BSS-machine. Therefore, if all activation, attention and pooling functions as well as the input and positional embedding are computable by BSS-machines, then the entire transformer is as well.

Because arithmetic circuits without sign gates compute only polynomial functions, they can be evaluated by a BSS-machine in a single step. To add a sign gate, we can use the polynomial that defines the value of its predecessor and use two branching states to determine its output value. Then we can continue the computation for the remainder of the circuit by replacing the sign gate with a constant gate for 1, 0 or -1.

Unique hard attention can also be computed on BSS-machines by simply finding the first maximal attention score and selecting the corresponding vector. To simulate a transformer, we first compute the input and positional embedding for each input vector and add them together. Then, for each layer and each vector, we compute the attention scores for all vectors, find the maximum and compute the activation function. This algorithmic description results in the following theorem:

Theorem 3.1. *Let R be any ordered ring. Then all functions in* $\mathbf{R\text{-}FT}_R\big(\mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{WS}_R(\mathrm{left})\big)$ *are computable by* $\mathbf{BSS}_R$*-machines.*

If we replace the arithmetic circuits with feedforward networks, we can also perform the simulation on a linear BSS-machine. Such a machine can compute the output of a feedforward network by multiplying values with the fixed weights and computing ReLU with a branching state.

We can allow the usage of dot-products for the attention functions by evaluating them in the branching states for finding the maximal score. So instead of two attention scores, the branching polynomials take two vectors as an input. Since dot-products are polynomial of degree 2, we also get:

Theorem 3.2. *Let R be any ordered ring. Then all functions in* $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{FF}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *are computable by linear* $\mathbf{BSS}_R$*-machines with degree* 2 *branching.*

Both of these theorems can be easily extended to **S-$\mathfrak{C}$** instead of **R-$\mathfrak{C}$**.

4 Simulating BSS-Machines with Circuit Transformers

We encode the contents of the tape as a sequence of real vectors each representing a single cell. Then we construct a transformer that is able to simulate a single computation step of the machine on this sequence. A vector $\overline{x}$ within the transformer consists of the following components:

x_v	the value of the cell,	x_s	encodes the state of the machine,
x_p	the positional embedding,	x_{one}	is always set to 1,
x_c	the cell's position relative to the head,	x_{trim}	for indicating if $\overline{x}$ is part of the output,
x_f	a flag used during attention and	x_{hlt}	for indicating a halting state.

We can use an arithmetic circuit with sign gates to perform a single step of computation for one cell in this representation. The values of x_v, x_c and x_s are then passed to the next iteration of the transformer and the others can be reconstructed from those. This description is a simplyfied version of the full construction, which can be found in the appendix.

4.1 Performing a Step of Computation

A single step in the computation of a BSS-machine is determined by its state, which we encode in x_s using distinct numbers. The input state always gets code 0, while the others are arbitrary. We can use sign gates to compare x_s to these finitely many possible encodings of states and perform different operations based on that. In all cases, x_s is set to the encoding of the next state.

- For shift states, we add ± 1 to x_c, which moves all cells relative to the head.
- For computation states, we evaluate the corresponding polynomial map $p\colon R^m \to R^n$ with the values x_v from the cells with $x_c \in \{1, \dots, m\}$. The resulting values are then stored in the vectors with positions $x_c \in \{1, \dots, n\}$.
- For branching states, we evaluate the corresponding polynomial in the same way and use the sign of the result to choose the next state.

4.2 Transferring Values between Vectors

The input embedding takes in the values of x_v, x_c and x_s, while x_p, x_f and x_{hlt} are set to 0. We can also use it to set $x_{one} = 1$. In each step, we only need to transfer values from a constant number of vectors, the ones with positions $x_c \in \{1, \dots, m\}$. Therefore, we can use a sign gate to set the flag x_f when x_c is the desired position and the attention function $f_f^{\mathrm{att}}(\overline{x}, \overline{y}) = x_{one} \cdot y_f$ to have all vectors attend to exactly the right vector.

The final step to make this computation work is handling the edges of the tape. By our definition of transformers, a new zero vector is appended to the sequence before every step. It only gets non-zero values in x_p and x_{one}. Because it always has the largest positional embedding among all vectors, we can use the attention function $f_p^{\mathrm{att}}(\overline{x}, \overline{y}) = x_{one} \cdot y_p$ to attend to this vector. Then we use a sign gate to compute the correct x_c only if x_p is the maximal positional embedding. If the current state is a shift state to the right, we want the new vector to represent a cell to the left of the tape. In all other cases, it can represent a cell to the right of the tape. Similar to the positional embedding, we can attend to the vectors with maximal and minimal cell positions and add or subtract 1 depending on the current state. The value x_v for the new cell remains 0, while x_s can be copied from a previous vector.

If the input sequence is shorter than the input of the function for any computation state, the transfer of this input might not work. This can be checked by computing the maximal cell position and therefore mitigated by delaying the computation while appending the new vector to the end of the tape.

4.3 Removing the Sign Gates

Because the result of hard attention can be non-continuous, we can use it to compute the sign function without sign gates. To do this, we add an extra dimension x_{sign} to the vectors which is also passed between iterations. Initially, it is zero for all vectors and it will always be restricted to the values $\{0, 1\}$. We can use the attention function $f^{\mathrm{att}}_{\mathrm{sign}}(\overline{x}, \overline{y}) = x_{one} \cdot y_{\mathrm{sign}}$ to compute the maximum y_{sign} among all vectors. If it is 0, we set $x_{\mathrm{sign}} = 1$ everywhere and skip the computation step. If it is 1, we continue as usual. This can be achieved by multiplying the respective values by either y_{sign} or $1 - y_{\mathrm{sign}}$. Therefore, one additional iteration of the transformer might be required for the computation, but from the second iteration onwards, there is always at least one vector with $x_{\mathrm{sign}} = 1$ and at least one vector with $x_{\mathrm{sign}} = 0$, which includes the appended one.

This setup allows us to compute the characteristic functions of comparison operators using an attention head. Suppose we need to compute $\mathrm{sign}(x_v)$ for each vector $\overline{x}$. Within an attention head using the attention function $f^{\mathrm{att}}(\overline{x}, \overline{y}) = x_v \cdot y_{\mathrm{sign}}$,

- if $x_v > 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\mathrm{sign}} = 1$,
- if $x_v < 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\mathrm{sign}} = 0$ and
- if $x_v = 0$, all attention scores are 0, so $\overline{x}$ attends to the first vector in the sequence which always has $y_{\mathrm{sign}} = 1$.

Therefore the resulting y_{sign} is the value of the characteristic function of $\{x_v \geq 0\}$. A second attention head with $f^{\mathrm{att}}(\overline{x}, \overline{y}) = -x_v \cdot y_{\mathrm{sign}}$ can compute the characteristic function of $\{x_v \leq 0\}$ and these two can be combined to compute $\mathrm{sign}(x_v) = \chi_{\geq 0}(x_v) - \chi_{\leq 0}(x_v)$.

4.4 Simulating the Input and Output Maps

We described all necessary steps to perform the computation on the previously defined encoding, but the input in the first iteration does not adhere to this encoding. We also need to compute the input map $\mathcal{I}_\infty$, which encodes the length of the sequence on the tape and the output map $\mathcal{O}_\infty$ which extracts the output values from the tape.

As $\mathcal{I}_\infty$ doubles the number of non-zero cells, the transformer needs additional iterations to add enough vectors to the sequence. We can use the additional iteration for the sign gates to compute the input length n, store it in all vectors and delay the actual computation, until the number of vectors is exactly $2 \cdot n$. If $n > 0$, the new vectors get a value of $x_v = 1$. Meanwhile, we add the cell positions to all vectors by setting

$$x_c = \begin{cases} x_p & \text{if } x_p \leq n \\ x_p - 2n & \text{otherwise} \end{cases}.$$

To simulate $\mathcal{O}_\infty$, when z_{out} is reached, we can set the flag $x_f = x_c + m$ for all vectors with $x_c \leq 0$ and $x_v = 0$, where m is the total number of vectors. Then we use the attention function $f^{\mathrm{att}}(\overline{x}, \overline{y}) = x_{one} \cdot y_f$, to find the rightmost such cell and compute the length of the output sequence $n = -x_c$ from its cell position. The output values are extracted one at a time by having the new vector attend to the next cell position in $1, \ldots, n$ using another flag. For these vectors, we also set $x_{\mathrm{trim}} = 1$ and once all n vectors are copied, $x_{hlt} = 1$. When putting all of this together, we get:

Theorem 4.1. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\textbf{R-FT}_R\big(\textbf{FC}\textsc{irc}_R, \textbf{IPE}_R, \textbf{FC}\textsc{irc}_R, \textbf{DPA}_R, \textbf{WS}_R(\mathrm{left})\big)$ *contains all* $\textbf{BSS}_R$*-computable functions (without division).*

4.5 Simulating BSS-Machines using Chain of Thought

If we change the definition of recursion from **R-$\mathfrak{C}$** to **S-$\mathfrak{C}$**, a slightly more complex construction is required. A single computation step can be simulated in mostly the same way as before, but we can only temporarily change the values of previously added vectors. Therefore, instead of storing the position of each cell relative to the tape head, we need to store an absolute position for each cell and for the head. The head position is always the cell position of the second-to-last input vector, so this value can be copied to the last vector. To avoid vectors with outdated values while collecting previous values, we need to choose the one with the highest positional embedding among those matching the

desired cell position. We can compute the maximal positional embedding n and define the attention function $f^{\text{att}}(\overline{x}, \overline{y}) = n \cdot y_f + x_{one} \cdot y_p$ This requires storing the value n in $\overline{x}$ beforehand.

As only the values of the last vector are preserved, the result of the polynomial function of a computation state must be stored entirely in that vector. But, in a BSS-machine, those values should be stored in multiple tape cells. So while looking up the value for a specific position, we also need to check a constant number of cells to the left of it and choose the vector with the highest positional embedding among all those. Then the computation step can be performed as before, but for shift states, we now step x_c in the opposite direction and for any state thats not a computation state, we need to perform the same value transfers to the last vector to keep the representation consistent.

The computation of the sign function also works in the same way, but the first vector can now have $x_{\text{sign}} = 0$ instead of 1. In that case, the attention head computes $\chi_{>0}$, which needs to be accounted for. The simulation of $\mathcal{I}_\infty$ now requires $3 \cdot n$ vectors, because the values of the first n need to be copied one at a time. This happens after the cells with value 1 are added. Finally, the simulation of the output length cannot be done in the same way as before, because zero values in vectors can be invalidated by later vectors. Instead we find the first zero to the left of the head by repeatedly performing right shift operations. This takes a linear amount of steps with respect to the output length. By working around all these limitations, we get the same theorem for **S-$\mathfrak{C}$**.

Theorem 4.2. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\textbf{S-FT}_R\big(\textbf{FCIRC}_R, \textbf{IPE}_R, \textbf{FCIRC}_R, \textbf{DPA}_R, \textbf{WS}_R(\mathrm{left})\big)$ *contains all* $\textbf{BSS}_R$*-computable functions (without division).*

5 Simulating BSS-Machines using Feedforward Networks

The main difference between arithmetic circuits and feedforward networks is the restriction on multiplication. Even if we can compute the sign function with attention, we cannot multiply its result by an arbitrary value to perform conditional computation. However, as feedforward networks have access to the ReLU function, we can simulate these multiplications in a different way. This construction makes use of the following Lemma.

Lemma 5.1. *Let R be an ordered ring and $f\colon R^* \to R^*$ such that $\sum_{i=1}^{|X|} |f(X)_i| \leq 1$ for all X. Then for all $T \in \textbf{FT}_R\big(A, B, \textbf{FF}_R, C, \textbf{WS}_R(f)\big)$, where A, B and C are arbitrary, we get $\max\{\|T(X)_i\| \mid 1 \leq i \leq |X|\} \in \mathcal{O}(\max\{\|f^{in}(X_i) + f^{pos}(i,n)\| \mid 1 \leq i \leq |X|\})$. Here $\|\overline{x}\| = \max\{|x_i| \mid 1 \leq i \leq |\overline{x}|\}$ denotes the maximum norm.*

5.1 Simulating Linear BSS-Machines using Circuit Attention

Firstly, we set $x_{one} = 1$ and use one attention function $f^{\text{att}}(\overline{x}, \overline{y}) = x_{one} \cdot y_i$ for each coordinate y_i to compute a vector $\overline{x}^{\max}$ which contains the maximal values among all vectors. Similarly, we compute $\overline{x}^{\min}$ containing the minimal values. Due to Lemma 5.1, there is a constant $c \in R$ such that all intermediate values in the feedforward networks of T are bounded by $G = c \cdot \max\{|x_i^{\max}|, |x_i^{\min}| \mid 1 \leq i \leq d\}$. We perform a construction similarly to Section 4.3, but we set $x_{\text{sign}} = G$ instead of 1 in the first iteration. This ensures, that $G = c \cdot x_{\text{sign}}$ in the next iteration, so by multiplying x_{sign} by c we get $G = x_{\text{sign}}$ for all but the last vector. Let $\hat{y}_{\text{sign}}$ be the maximum y_{sign} among all vectors. Then

$$\mathrm{ReLU}(a + \hat{y}_{\text{sign}} - G) = \begin{cases} 0 & \text{if } \hat{y}_{\text{sign}} = 0 \\ a & \text{if } \hat{y}_{\text{sign}} = G \end{cases} \quad \text{and} \quad \mathrm{ReLU}(b - \hat{y}_{\text{sign}}) = \begin{cases} b & \text{if } \hat{y}_{\text{sign}} = 0 \\ 0 & \text{if } \hat{y}_{\text{sign}} = G \end{cases}$$

for all $0 \leq a, b \leq G$. These functions are computable by feedforward networks and we can use them to perform a different computation in the first iteration before x_{sign} is setup correctly.

In all following iterations, this setup allows for conditional computation based on the sign function. Suppose we need to compute either a or b based on $\mathrm{sign}(x_v)$ for each vector $\overline{x}$. Within an attention head with $f^{\text{att}}(\overline{x}, \overline{y}) = x_v \cdot y_{\text{sign}}$,

- if $x_v > 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\text{sign}} = G$,
- if $x_v < 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\text{sign}} = 0$ and
- if $x_v = 0$, all attention scores are 0, so $\overline{x}$ attends to the first vector in the sequence which always has $y_{\text{sign}} = G$.

Therefore, for all $0 \leq a, b \leq G$, we get

$$\mathrm{ReLU}(a + y_{\mathrm{sign}} - G) = \begin{cases} a & \text{if } x_v \geq 0 \\ 0 & \text{if } x_v < 0 \end{cases} \quad \text{and} \quad \mathrm{ReLU}(b - y_{\mathrm{sign}}) = \begin{cases} 0 & \text{if } x_v \geq 0 \\ b & \text{if } x_v < 0 \end{cases}.$$

The sum of these two is the desired conditional function and is computable by a feedforward network. This allows us to perform computations based on the machine state and the transfer of values via attention then works in the same way as before. The computation state polynomials have degree 1 and can be computed by feedforward networks. For a branching state using a polynomial p, we can use the attention function $f^{\mathrm{att}}(\overline{x}, \overline{y}) = p(x_{v1}, x_{v2}, \dots) \cdot y_{\mathrm{sign}} \in \mathbf{FCIRC}_R$ to conditionally compute the next state based on $\mathrm{sign}(p(x_{v1}, x_{v2}, \dots))$.

Theorem 5.2. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{FCIRC}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *contains all functions computable by linear* $\mathbf{BSS}_R$*-machines.*

5.2 Simulating Degree 2 Branching using Dot-Product Attention

All the attention functions used in the previous sections lie in $\mathbf{DPA}_R$ except for those that compute the branching polynomials. However, if those polynomials have a maximal degree of 2, a slightly different construction allows us to simulate them as well.

Let $p(x_1, \dots, x_m) = \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m} b_i x_i + c$. Using our previous construction we can transfer the values from cell positions $1, \dots, m$ to all vectors. Let $u_1, \dots, u_m$ be those values. For each vector $\overline{x}$ and all $i \in \{1, \dots, m\}$, we set the additional dimensions $x_{vi} = u_i$ and $x_{bi} = \mathrm{ReLU}(u_i + x_{\mathrm{sign}} - G) = \frac{x_{\mathrm{sign}}}{G} \cdot u_i$. We also need $x_{bsg} = \mathrm{ReLU}(1 + x_{\mathrm{sign}} - G) = \frac{x_{\mathrm{sign}}}{G}$. Then the attention function

$$f^{\mathrm{att}}(\overline{x}, \overline{y}) = \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_{vi} y_{bj} + \sum_{i=1}^{m} b_i x_{vi} y_{bsg} + c x_{one} y_{bsg}$$

can be used to compute the branching condition, because we get

$$f^{\mathrm{att}}(\overline{x}, \overline{y}) = p(u_1, \dots, u_m) \cdot y_{bsg} = p(u_1, \dots, u_m) \cdot \frac{y_{\mathrm{sign}}}{G}.$$

Theorem 5.3. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *contains all functions computable by linear* $\mathbf{BSS}_R$*-machines with degree 2 branching.*

The construction from Section 4.5 is also possible within these constraints. Therefore, the previous two theorems also hold for $\mathbf{S\text{-}}\mathfrak{C}$ instead of $\mathbf{R\text{-}}\mathfrak{C}$.

6 Conclusion

Theorems 3.1 and 4.1 establish an equality between the functions computed by $\mathbf{BSS}_R$-machines and $\mathbf{R\text{-}FT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ for any $\mathbb{Z}$-ordered ring R. Theorems 3.2 and 5.2 do the same for linear $\mathbf{BSS}_R$-machines with degree-2 branching and $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$. These results also hold when replacing $\mathbf{R\text{-}}\mathfrak{C}$ with $\mathbf{S\text{-}}\mathfrak{C}$. Table 1 gives a summary of our results.

Time Complexity Our theorems only describe the relation of computability between the two models, but the proofs also give upper bounds on the time complexity of the simulations. As the transformers can simulate a computation step in a single iteration, they can simulate a BSS-machine that runs in time $t(n)$ in $t(n) + O(n)$ iterations. The linear overhead comes from the simulation of the input and output maps. On the other hand, a BSS-machine can simulate a single transformer iteration in quadratic time. Therefore, the classes of functions computable in polynomial time and above are equal between BSS-machines and recursive hard attention transformers. This is true for both equivalences. The full statements and proofs can be found in the appendix.

Theorem 6.1 (Sketch)**.** *Let R be any $\mathbb{Z}$-ordered ring. Then the classes of functions computable in polynomial time by* $\mathbf{BSS}_R$*-machines and transformers in* $\mathbf{GT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *are equal.*

Corollary 6.2 (Sketch)**.** *Let R be any $\mathbb{Z}$-ordered ring. Then the classes of functions computable in polynomial time by linear* $\mathbf{BSS}_R$*-machines with degree-2 branching and transformers in* $\mathbf{GT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *are equal.*

Table 1: Summary of Results excluding the statements about time complexity

Ring	Machine	Branching		Rec	PE	Activation	Attention	Pooling
ordered	poly	poly	$\supseteq$	**R-FT**		$\mathbf{FC}\mathbf{IRC}_R[\mathrm{sign}]$		left / right
ordered	linear	degree-2	$\supseteq$	**R-FT**	$\mathbf{FF}_R$	$\mathbf{FF}_R$	$\mathbf{DPA}_R$	left / right
$\mathbb{Z}$-ord.	poly	poly	$=$	**R-FT**	$\mathbf{IPE}_R$	$\mathbf{FC}\mathbf{IRC}_R$	$\mathbf{DPA}_R$	left / right
$\mathbb{Z}$-ord.	poly	poly	$=$	**S-FT**	$\mathbf{IPE}_R$	$\mathbf{FC}\mathbf{IRC}_R$	$\mathbf{DPA}_R$	left / right
$\mathbb{Z}$-ord.	linear	degree-2	$=$	**R-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{DPA}_R$	left / right
$\mathbb{Z}$-ord.	linear	degree-2	$=$	**S-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{DPA}_R$	left / right
$\mathbb{Z}$-ord.	linear	poly	$\subseteq$	**R-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{FC}\mathbf{IRC}_R$	left / right
$\mathbb{Z}$-ord.	linear	poly	$\subseteq\subseteq$	**S-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{FC}\mathbf{IRC}_R$	left / right
$\mathbb{Z}$-ord.	poly	poly	$\subseteq\subseteq$	**R-FT**	$\mathbf{IPE}_R$	$\mathbf{FC}\mathbf{IRC}_R$	$\mathbf{DPA}_R$	average
$\mathbb{Z}$-ord.	poly	poly	$\subseteq\subseteq$	**S-FT**	$\mathbf{IPE}_R$	$\mathbf{FC}\mathbf{IRC}_R$	$\mathbf{DPA}_R$	average
$\mathbb{Z}$-ord.	linear	degree-2	$\subseteq\subseteq$	**R-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{DPA}_R$	average
$\mathbb{Z}$-ord.	linear	degree-2	$\subseteq\subseteq$	**S-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{DPA}_R$	average
$\mathbb{Z}$-ord.	linear	poly	$\subseteq\subseteq$	**R-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{FC}\mathbf{IRC}_R$	average
$\mathbb{Z}$-ord.	linear	poly	$\subseteq\subseteq$	**S-FT**	$\mathbf{IPE}_R$	$\mathbf{FF}_R$	$\mathbf{FC}\mathbf{IRC}_R$	average

Average Attention Average attention, where all vectors that produce maximal attention scores are weighted equally to compute the result, can be used to simulate leftmost attention for our constructions. Some attention heads avoid ties entirely. For the integer-valued attention functions, we can replace the factors of x_{one} with the sequence length and add the term $-x_{one} \cdot y_p$ to simulate leftmost attention. The construction for sign can be fixed by picking two vectors with different x_{sign} and raising their attention scores by the same amount.

Therefore, average attention transformers can simulate BSS-machines. But computing average attention on a BSS-machine requires a division operation. A $\mathbf{BSS}_{\mathbb{R}}$-machine with division is able to compute reciprocals of real numbers, while average attention only uses reciprocals of natural numbers. So an equivalence result for average attention would require either adding true division to the transformers or somehow restricting division for the BSS-machines.

Hovewer, if we restrict our view to decision problems, leftmost and average attention are equivalent, because division can be eliminated from BSS-machines by using a numerator-denominator representation for all values, that are not part of the output.

Further Research One closely related open question is, whether one could obtain similar results for soft attention transformers. Using the softmax function requires real values and it makes it difficult to attend to one specific vector. It also makes the entire transformer a continuous function, preventing the direct computation of sign. However, when reintroducing it to the activation functions, we think that such transformers could simulate or approximate certain kinds of BSS-machines.

References

Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir V. Podolskii. Logical languages accepted by transformer encoders with hard attention. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=gbrHZq07mq`.

Timon Barlag, Vivian Holzapfel, Laura Strieker, Jonni Virtema, and Heribert Vollmer. Graph neural networks and arithmetic circuits. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL `http://papers.nips.cc/paper_files/paper/2024/hash/09bf6a87e80d099cf17c6347301c6120-Abstract-Conference.html`.

Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: Np-completeness, recursive functions and universal machines. *Bulletin of the*

American Mathematical Society, 21(1):1–46, 1989. ISSN 1088-9485. doi: https://doi.org/10.1090/S0273-0979-1989-15750-9.

Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Computation over a Ring*, pages 69–81. Springer New York, 1998. ISBN 978-1-4612-0701-6. doi: 10.1007/978-1-4612-0701-6_3. URL https://doi.org/10.1007/978-1-4612-0701-6_3.

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, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 5544–5562. PMLR, 2023. URL https://proceedings.mlr.press/v202/chiang23a.html.

Lena Ehrmuth and Laura Strieker. Average attention transformers and arithmetic circuits, 2026. URL https://arxiv.org/abs/2605.04683.

Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810, 2022. doi: 10.1162/tacl_a_00490. URL https://aclanthology.org/2022.tacl-1.46.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. doi: 10.1162/tacl_a_00562. URL https://aclanthology.org/2023.tacl-1.31.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing-complete. *J. Mach. Learn. Res.*, 22:75:1–75:35, 2021. URL http://jmlr.org/papers/v22/20-302.html.

H. T. Siegelmann and E. D. Sontag. On the Computational Power of Neural Nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995. ISSN 0022-0000. doi: 10.1006/jcss.1995.1013. URL https://www.sciencedirect.com/science/article/pii/S0022000085710136.

Lena Strobl. Average-hard attention transformers are constant-depth uniform threshold circuits, 2023. URL https://arxiv.org/abs/2308.03212.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. doi: 10.1162/tacl_a_00663. URL https://aclanthology.org/2024.tacl-1.30.

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.

Heribert Vollmer. *Arithmetic Circuits*, pages 173–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-662-03927-4. doi: 10.1007/978-3-662-03927-4_6. URL https://doi.org/10.1007/978-3-662-03927-4_6.

Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/13d7f172259b11b230cc5da8768abc5f-Abstract-Conference.html.

A Proofs of Section 3

Lemma A.1. *Let R be a ring and $f \in \mathbf{FC}_{\mathbf{IRC}}{}_R[\mathrm{sign}]$. Then f can be computed by a $\mathbf{BSS}_R$-machine.*

Proof. Since any circuit without sign gates computes a polynomial function, a BSS-machine can compute its output in a single computation state. For a circuit with sign gates, let g be the first sign gate in order of any topological sorting of the circuit. The subcircuit for g contains all gates g' such that there is a path from g' to g. Since this subcircuit contains no sign gates, a BSS-machine can compute the value b of the predecessor of g. Then, using two branching states it can decide whether $b > 0$, $b = 0$ or $b < 0$. We can then replace g with a constant gate containing either 1, 0 or -1. The resulting circuit has one fewer sign gate than the original one. By induction, a BSS-machine can compute the output of circuits with any number of sign gates. □

Theorem 3.1. *Let R be any ordered ring. Then all functions in $\mathbf{R\text{-}FT}_R\big(\mathbf{FC}_{\mathbf{IRC}}{}_R[\mathrm{sign}], \mathbf{FC}_{\mathbf{IRC}}{}_R[\mathrm{sign}], \mathbf{FC}_{\mathbf{IRC}}{}_R[\mathrm{sign}], \mathbf{FC}_{\mathbf{IRC}}{}_R[\mathrm{sign}], \mathbf{WS}_R(\mathrm{left})\big)$ are computable by $\mathbf{BSS}_R$-machines.*

Proof. Let $(R, d, R^m, f^{\text{in}}, f^{\text{pos}}, f^{\text{att}}_{1,1}, f^{\text{att}}_{1,2}, \ldots, f^{\text{att}}_{K,H}, f^{\text{pool}}, \ldots, f^{\text{pool}}, f^{\text{act}}_1, \ldots, f^{\text{act}}_K)$ be a transformer with $f^{\text{pool}} \in \mathbf{WS}_R(\mathrm{left})$. We give Algorithm 1 which computes its output while using f^{in}, f^{pos}, the attention functions and the activation functions as subroutines. Due to Lemma A.1, all of those are computable by $\mathbf{BSS}_R$-machines. Therefore, the algorithm can be executed by a $\mathbf{BSS}_R$-machine. The exact translation into states is not described here. It can then be applied recursively while checking whether $x_{hlt} \neq 0$. Finally, input and output projections are also computable. □

Algorithm 1 computes the output of a transformer using leftmost hard attention.

procedure $T(X \in (R^m)^*)$
  Allocate arrays $\mathrm{vectors}$ and $\mathrm{new_vectors}$ of length $|X| + 1$ for R^d-vectors
  Allocate an array $\mathrm{att_results}$ of constant length H for R^d-vectors
  for $i \leftarrow 1, \ldots, |X|$ **do**
    $\mathrm{vectors}[i] \leftarrow f^{\text{in}}(X[i]) + f^{\text{pos}}(i, |X| + 1)$
  end for
  $\mathrm{vectors}[|X| + 1] \leftarrow f^{\text{in}}(0) + f^{\text{pos}}(|X| + 1, |X| + 1)$
  for $k \leftarrow 1, \ldots, K$ **do**
    for $i \leftarrow 1, \ldots, |X|$ **do**
      for $h \leftarrow 1, \ldots, H$ **do**
        $\mathrm{att_results}[h] \leftarrow \mathrm{vectors}[1]$
        for $j \leftarrow 2, \ldots, |X|$ **do**
          if $f^{\text{att}}_{k,h}(\mathrm{vectors}[i], \mathrm{vectors}[j]) > f^{\text{att}}_{k,h}(\mathrm{vectors}[i], \mathrm{att_results}[h])$ **then**
            $\mathrm{att_results}[h] \leftarrow \mathrm{vectors}[j]$
          end if
        end for
      end for
      $\mathrm{new_vectors}[i] \leftarrow f^{\text{act}}_k(\mathrm{vectors}[i], \mathrm{att_results}[1], \ldots, \mathrm{att_results}[H])$
    end for
    $\mathrm{vectors} \leftarrow \mathrm{new_vectors}$
  end for
end procedure

Lemma A.2. *Let R be an ordered ring and $f \in \mathbf{FF}_R$. Then f can be computed by a linear $\mathbf{BSS}_R$-machine with degree 1 branching.*

Proof. A feedforward network $f\colon R^m \to R^n$ is a composition of affine transformations and the ReLU function. Any affine transformation can be used as the associated polynomial of a computation state and $\mathrm{ReLU}(x_1)$ can be computed by setting $x_1 = 0$ if $x_1 \leq 0$. □

Theorem 3.2. *Let R be any ordered ring. Then all functions in $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{FF}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ are computable by linear $\mathbf{BSS}_R$-machines with degree 2 branching.*

Proof. When f^{in}, f^{pos} and the activation functions are computable by feedforward networks, we can execute Algorithm 1 on a linear $\mathbf{BSS}_R$-machine. Those functions can be computed due to Lemma A.2, but a problem occurs with the if statement in line 13. However, notice that for all $\overline{x}, \overline{y}, \overline{z}$ and f^{att},

$$f^{\text{att}}(\overline{x}, \overline{y}) > f^{\text{att}}(\overline{x}, \overline{z}) \quad \Leftrightarrow \quad f^{\text{att}}(\overline{x}, \overline{y}) - f^{\text{att}}(\overline{x}, \overline{z}) > 0.$$

If $f^{\text{att}} \in \mathbf{DPA}_R$, then $f^{\text{att}}(\overline{x}, \overline{y}) - f^{\text{att}}(\overline{x}, \overline{z})$ is a polynomial of degree 2, so $f^{\text{att}}(\overline{x}, \overline{y}) - f^{\text{att}}(\overline{x}, \overline{z}) > 0$ can be decided with degree 2 branching. □

B Proofs of Section 4

Definition B.1. Let M be a $\mathbf{BSS}_R$-machine with state set $Z = \{z_{in}, z_{out}\} \cup Z_{\text{comp}} \cup Z_{\text{left}} \cup Z_{\text{right}} \cup Z_{\text{branch}}$, such that Z_{comp} contains only compute states, Z_{left} contains only left shift states, Z_{right} contains only right shift states and Z_{branch} contains only branch states. We use $s(z)$ to denote the unique successor of state z if it exists and $s_0(z), s_1(z)$ for $z \in Z_{\text{branch}}$. Let $\hat{m}$ be the maximum of all input and output dimensions of polynomials used by M.

We describe the transformer $T_{RC}(M)$ using four activation functions, but it actually has eight layers. Our activation functions use $\chi_{\geq}$ and $\chi_{=}$, which must be computed by an attention head as follows. To compute $\chi_{\geq}(a, b)$ within an activation function f^{act}, we use two layers both using the described attention heads. The first layers' activation function then stores the value $a - b$ in an additional coordinate $x_{\geq}$ for all vectors. The second layer has an additional attention head using $f^{\text{att}}(\overline{x}, \overline{y}) = x_{\geq} \cdot y_{\text{sign}}$ to compute $\chi_{\geq}(a, b)$. It can then apply f^{act} with $\chi_{\geq}(a, b)$ replaced by y_{sign} of that attention result. This process can be used in parallel for each occurence of $\chi_{\geq}$. We also use $G = 1$, $a \otimes b = a \cdot b$ and $\text{cond}\,(c, a, b) = c \otimes a + (G - c) \otimes b$. Then $\chi_{=}$ can be defined as $\chi_{=}(a, b) = \chi_{\geq}(a, b) \otimes \chi_{\geq}(b, a)$.

The input function f^{in} is defined as $f^{\text{in}}(\overline{x})_v = x_v$, $f^{\text{in}}(\overline{x})_c = x_c$, $f^{\text{in}}(\overline{x})_s = x_s$, $f^{\text{in}}(\overline{x})_{\text{sign}} = x_{\text{sign}}$, $f^{\text{in}}(\overline{x})_{len} = x_{len}$, $f^{\text{in}}(\overline{x})_{\text{trim}} = x_{\text{trim}}$, $f^{\text{in}}(\overline{x})_{one} = 1$ and zero in every other coordinate.

The first layer uses one attention head with the attention function $f^{\text{att}}_p(\overline{x}, \overline{y}) = x_{one} \cdot y_p$. Its actvation function sets $f^{\text{act}}_1(\overline{x}, \overline{y}^p)_{end} = \chi_{\geq}(2 \cdot x_p + 1, 2 \cdot y^p_p)$. All coordinates, which are not described here, are retained from $\overline{x}$.

The second layer uses the attention functions $f^{\text{att}}_{\text{sign}}(\overline{x}, \overline{y}) = x_{one} \cdot y_{\text{sign}}$, $f^{\text{att}}_{+c}(\overline{x}, \overline{y}) = x_{one} \cdot y_c$, $f^{\text{att}}_{-c}(\overline{x}, \overline{y}) = -x_{one} \cdot y_c$, $f^{\text{att}}_{prev}(\overline{x}, \overline{y}) = x_{one} \cdot y_p - x_{one} \cdot y_{end}$ and f^{att}_p.

$$\begin{aligned}
f^{\text{act}}_2(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{+c}, \overline{y}^{-c}, \overline{y}^{prev}, \overline{y}^p) &= \overline{u} \\
c_1 &= x_c + \chi_{=}(x_p, y^p_p) \otimes \\
&\quad \left(y^{+c}_c + 1 + \sum_{z \in Z_{\text{right}}} \chi_{=}(y^{prev}_s, \langle z \rangle) \otimes \left(\chi_{\geq}(y^p_p - 1, y^{-c}_p) \otimes (y^{-c}_c - 1) - y^{+c}_c - 1 \right) \right) \\
u_c &= c_1 + \chi_{=}(y^{prev}_s, \langle z_{in} \rangle) \otimes \\
&\quad \left(x_p - y^{prev}_{len} - c_1 - \chi_{\geq}(2 \cdot y^{prev}_{len}, x_p) \otimes y^{prev}_{len} + \chi_{\geq}(y^{prev}_{len}, x_p) \otimes (2 \cdot y^{prev}_{len}) \right) \\
u_v &= x_v + \chi_{=}(x_s, \langle z_{in} \rangle) \otimes \text{cond}\left(\chi_{\geq}(y^{prev}_{len}, x_p), 0, \chi_{\geq}(2 \cdot y^{prev}_{len}, x_p) \otimes (1 - x_v) \right) \\
u_s &= x_s + \chi_{=}(x_p, y^p_p) \otimes y^{prev}_s \\
u_{len} &= \text{cond}\left(y^{\text{sign}}_{\text{sign}}, \chi_{=}(x_p, y^p_p) \otimes y^{prev}_{len} + x_{len}, y^p_p - 1 \right) \\
u_f &= \chi_{=}(x_v, 0) \otimes \chi_{\geq}(0, u_c) \otimes (u_c + y^p_p) \\
u_{fi} &= \chi_{=}(u_c, i) \quad \text{for } i = 1, \ldots, \hat{m} \\
u_{out} &= \text{cond}\left(\chi_{=}(y^{prev}_{\text{trim}}, 0), \chi_{=}(u_c, 1), \chi_{=}(u_c, y^{prev}_c + 1) \right)
\end{aligned}$$

The third layer uses the attention functions f^{att}_{-c}, $f^{\text{att}}_{fi} = x_{one} \cdot y_{fi}$ for $i = 1, \ldots, \hat{m}$ and $f^{\text{att}}_{f}(\overline{x}, \overline{y}) = x_{one} \cdot y_f$.

$$
\begin{aligned}
f^{\text{act}}_3(\overline{x}, \overline{y}^{-c}, \overline{y}^1, \ldots, \overline{y}^{\hat{m}}, \overline{y}^f) &= \overline{v} \\
v_v &= x_v + \sum_{z \in Z_{\text{comp}}} \chi_{=}(x_s, \langle z \rangle) \otimes \sum_{i=1}^{n} \chi_{=}(x_c, i) \otimes \left(p_z(y^1_v, \ldots, y^m_v)_i - x_v\right) \\
v_{vari} &= y^i_v \quad \text{for } i = 1, \ldots, \hat{m} \\
v_c &= x_c - \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z \rangle) \otimes 1 + \sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z \rangle) \otimes 1 \\
v_{len} &= x_{len} + \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \\
&\qquad \Big(\text{cond}\left(\chi_{=}(y^f_v, 0) \otimes \chi_{\geq}(0, y^f_c), -y^f_c, \chi_{\geq}(0, y^{-c}_c) \otimes (1 - y^{-c}_c)\right) - x_{len} \Big)
\end{aligned}
$$

The fourth layer uses the attention functions $f^{\text{att}}_{\text{sign}}$, f^{att}_{+c}, $f^{\text{att}}_{out}(\overline{x}, \overline{y}) = x_{one} \cdot y_{out}$ and f^{att}_{p}.

$$
\begin{aligned}
f^{\text{act}}_4(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{+c}, \overline{y}^{out}, \overline{y}^{p}) &= \overline{w} \\
nis &= \text{cond}\left(\chi_{\geq}(y^{+c}_c, \hat{m}), x_s + \sum_{z \in Z \setminus Z_{\text{branch}} \setminus \{z_{out}\}} \chi_{=}(x_s, \langle z \rangle) \otimes (\langle s(z) \rangle - x_s) \right. \\
&\qquad \left. + \sum_{z \in Z_{\text{branch}}} \chi_{=}(x_s, \langle z \rangle) \otimes \Big(\text{cond}\left(\chi_{\geq}(p_z(x_{var1}, \ldots, x_{varm}), 0), \langle s_1(z) \rangle, \langle s_0(z) \rangle\right) - x_s \Big), x_s \right) \\
w_s &= \text{cond}\left(y^{\text{sign}}_{\text{sign}}, \text{cond}\left(\chi_{=}(x_s, \langle z_{in} \rangle), \text{cond}\left(\chi_{\geq}(y^p_p, 2 \cdot y^p_{len}), nis, x_s\right), nis\right), x_s\right) \\
w_v &= x_v + \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{=}(x_p, y^p_p) \otimes \left(y^{out}_v - x_v\right) \\
w_c &= x_c + \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{=}(x_p, y^p_p) \otimes \left(y^{out}_c - x_c\right) \\
w_{\text{sign}} &= G \\
w_{\text{trim}} &= x_{\text{trim}} + \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{=}(x_p, y^p_p) \otimes \left(\text{cond}\left(\chi_{\geq}(0, x_{len}), x_{\text{trim}}, 1\right) - x_{\text{trim}}\right) \\
w_{hlt} &= \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{\geq}(y^{out}_c, x_{len})
\end{aligned}
$$

Lemma B.2. *Let R be a $\mathbb{Z}$-ordered ring and M be a* $\mathbf{BSS}_R$*-machine. If X is a sequence of vectors such that the first vector has $x_{\text{sign}} = 1$ and the set of x_{sign} in X is $\{1\}$ or $\{0, 1\}$, then*

$$
\chi_{\geq}(a, b) = \begin{cases} 1 & \text{if } a \geq b \\ 0 & \text{if } a < b \end{cases}
$$

for all $a, b \in R$ within all layers of $T_{RC}(M)(X)$, the first vector of $T_{RC}(M)(X)$ has $x_{\text{sign}} = 1$ and the set of x_{sign} in $T_{RC}(M)(X)$ is $\{1\}$.

Proof. After the input layer, the last vector always has $x_{\text{sign}} = 0$, so there is at least one vector with $x_{\text{sign}} = 0$. An attention head computing $\chi_{\geq}(a, b)$ for $\overline{x}$ uses $f^{\text{att}}(\overline{x}, \overline{y}) = x_{\geq} \cdot y_{\text{sign}}$ with $x_{\geq} = a - b$. Therefore,

- if $x_{\geq} > 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\text{sign}} = 1$,
- if $x_{\geq} < 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\text{sign}} = 0$ and
- if $x_{\geq} = 0$, all attention scores are 0, so $\overline{x}$ attends to the first vector which has $y_{\text{sign}} = 1$.

So the resulting y_{sign} is the correct value. As x_{sign} cannot change until the final layer, this works for all layers. In the final layer, we get $f^{\text{act}}_4(x, \overline{y}^{\text{sign}}, \ldots)_{\text{sign}} = G = 1$, so the set of x_{sign} is $\{1\}$. □

Theorem 4.1. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{R\text{-}FT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\text{left})\big)$ *contains all* $\mathbf{BSS}_R$*-computable functions (without division).*

The proof of this theorem uses a partial function Φ to define the following invariants which are necessary for the simulation to function:

- All vectors encode the same valid state.
- Each vector represents a different cell and there are no gaps between them.
- There is a cell for each position that can be read or written to (positions $1, \ldots, m$).
- The set of y_{sign} is $\{0, 1\}$ and the first vector has $y_{\text{sign}} = 1$.
- Every vector has $y_{\text{trim}} = 0$.
- Every vector has $y_{hlt} = 0$.

The proof is divided into three parts: One for the input state, one for the output state and one for every other state. During the simulation of the input state, we need to ensure that:

- The state does not change until all invariants above are met.
- All vectors store the original input length in y_{len}.
- The values of the original input are preserved.
- The $|X|$ cells to the left of the tape head are set to value 1.

For the other states, additional invariants depend on the state types:

- Values do not change except for computation states.
- Cell indices do not change except for shift states.

The simulation of the output state actually breaks the invariants for the cells, y_{trim} and eventually y_{hlt}. Instead, we need the following:

- All vectors store the output length in y_{len}.
- The vectors added in this state have cells $1, 2, \ldots$.
- Those vectors copy the values from the others with the same cell.
- They also have $y_{\text{trim}} = 1$, if the output length is not zero.
- All other vectors have $y_{\text{trim}} = 0$.
- Finally, $y_{hlt} = 0$, unless the number of added vectors is at least the output length.

Proof. Let M be any $\mathbf{BSS}_R$-machine using the state set Z. It defines a transition relation $\vdash \subseteq (Z \times R_\infty)^2$. We will define a partial function $\Phi\colon (R^d)^* \to Z \times R_\infty$ which maps the output sequences of the transformer $T = T_{RC}(M)$ to configurations of M. Then we prove that

1. For any $X \in R^*$, there exists a $t \in \mathbb{N}$, such that $(z_{in}, \mathcal{I}_\infty(X)) \vdash \Phi(T^t(p^*_{in}(X)))$.
2. For any $X \in (R^d)^*$ and $z \in Z \setminus \{z_{in}, z_{out}\}$, $\Phi(X) = (z, \mu)$ implies $\Phi(X) \vdash \Phi(T(X))$.
3. For any $X \in (R^d)^*$, $\Phi(X) = (z_{out}, \mu)$ implies that there exists a $t \in \mathbb{N}$, such that $x_{hlt} = 1$ in the last vector of $T^t(X)$ and $p^*_{out}(trim(T^t(X))) = \mathcal{O}_\infty(y)$.

We also need to ensure that $x_{hlt} = 0$ in all other situations.

For any vector sequence Y, if the set of y_s in Y is $\{\langle z \rangle\}$ for some $z \in Z$, the set of y_c in Y is $C = \{c_{\min}, c_{\min} + 1, \ldots, c_{\max} - 1, c_{\max}\}$, $|Y| = |C|$, $c_{\min} \leq 1$, $c_{\max} \geq \hat{m}$, the set of y_{sign} in Y is $\{0, 1\}$, the first vector has $y_{\text{sign}} = 1$ and $y_{\text{trim}} = 0$ everywhere, then we define

$$\Phi(Y) = \big(z, (\ldots, 0, v_{c_{\min}}, \ldots, v_0 . v_1, \ldots, v_{c_{\max}}, 0, \ldots)\big)$$

where v_i is the y_v of the unique vector with $y_c = i$.

1. For any $X \in R^*$, the result of the input layer in $T(X)$ is zero in every coordinate, except for x_v, x_p and x_{one}. Layer 2 uses the attention function f_p^{att} to compute $\overline{y}^p$, so $f_2^{\text{act}}(x, \overline{y}^{\text{sign}}, \dots, \overline{y}^p)_{len} =$

$$\text{cond}\left(y^{\text{sign}}_{\text{sign}}, \dots, y^p_p - 1\right) = \text{cond}\left(0, \dots, |X| + 1 - 1\right) = 0 \cdot (\dots) + (1 - 0) \cdot |X| = |X|$$

for all vectors. Since x_{sign} does not change until layer 4, we get $\chi_{\geq}(a, b) = \chi_{=}(a, b) = 0$ for all $a, b \in R$. Therefore, $f_3^{\text{act}}(x, \overline{y}^{\text{sign}}, \dots)_{len} = x_{len} + \chi_{=}(x_s, \langle z_{out}\rangle) \otimes (\dots) = x_{len} = |X|$. Since

$$\begin{aligned} f_4^{\text{act}}(\overline{v}, \dots)_v &= v_v + \chi_{=}(v_s, \langle z_{in}\rangle) \otimes (\dots) = v_v + 0 \cdot (\dots) = v_v \\ = f_3^{\text{act}}(\overline{u}, \dots)_v &= u_v + \sum_{z \in Z_{\text{comp}}} \chi_{=}(u_s, \langle z\rangle) \otimes (\dots) = u_v \\ = f_2^{\text{act}}(\overline{x}, \dots)_v &= x_v + \chi_{=}(x_s, \langle z_{in}\rangle) \otimes (\dots) = x_v, \end{aligned}$$

x_v does not change. After the final layer, we also have

$f_4^{\text{act}}(x, \overline{y}^{\text{sign}}, \dots)_s = \text{cond}\left(y^{\text{sign}}_{\text{sign}}, \dots, x_s\right) = \text{cond}\left(0, \dots, 0\right) = 0 \cdot (\dots) + (1 - 0) \cdot 0 = 0 = \langle z_{in}\rangle$,

$f_4^{\text{act}}(\overline{x}, \dots)_{hlt} = \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{\geq}(y^{out}_c, x_{len}) = 0$ and $f_4^{\text{act}}(x, \dots)_{\text{sign}} = G = 1$ for all vectors. Therefore, Lemma B.2 applies inductively to all following iterations.

During those iterations, $f_1^{\text{act}}(\overline{x}, \overline{y}^p)_{end} = 1$ for the last vector and 0 for all others. So f^{att}_{prev} makes all vectors attend to the second-to-last vector, which we call $\overline{y}^{prev}$. For x_{len}, we now get

$$\begin{aligned} f_2^{\text{act}}(\overline{x}, \dots, \overline{y}^{prev}, \overline{y}^p)_{len} &= \text{cond}\left(y^{\text{sign}}_{\text{sign}}, \chi_{=}(x_p, y^p_p) \otimes y^{prev}_{len} + x_{len}, y^p_p - 1\right) \\ &= \text{cond}\left(G, \chi_{=}(x_p, y^p_p) \otimes y^{prev}_{len} + x_{len}, y^p_p - 1\right) \\ &= \chi_{=}(x_p, y^p_p) \otimes |X| + x_{len} \\ &= \begin{cases} |X| + 0 & \text{for the last vector} \\ 0 + |X| & \text{for the others} \end{cases} \end{aligned}$$

and $f_3^{\text{act}}(\overline{x}, \overline{y}^{\text{sign}}, \dots)_{len} = x_{len} + \chi_{=}(x_s, \langle z_{out}\rangle) \otimes (\dots) = x_{len} + 0 \otimes (\dots) = x_{len}$. Therefore $x_{len} = |X|$ for all vectors. Similarly, we get

$$f_4^{\text{act}}(\overline{x}, \dots)_{\text{trim}} = x_{\text{trim}} + \chi_{=}(x_s, \langle z_{out}\rangle) \otimes (\dots) = x_{\text{trim}} = 0.$$

For x_s, we get $f_2^{\text{act}}(x, \dots, y^{prev}, y^p)_s = x_s + \chi_{=}(x_p, y^p_p) \otimes y^{prev}_s = 0 + \chi_{=}(x_p, y^p_p) \otimes 0 = 0 = \langle z_{in}\rangle$ for all vectors and because $f_4^{\text{act}}(x, \overline{y}^{\text{sign}}, \dots)_s =$

$$\begin{aligned} &\text{cond}\left(G, \text{cond}\left(\chi_{=}(\langle z_{in}\rangle, \langle z_{in}\rangle), \text{cond}\left(\chi_{\geq}(y^p_p, 2 \cdot y^p_{len}), nis, \langle z_{in}\rangle\right), nis\right), \langle z_{in}\rangle\right) \\ &= G \otimes \text{cond}\left(\chi_{\geq}(y^p_p, 2 \cdot y^p_{len}), nis, \langle z_{in}\rangle\right) + \langle z_{in}\rangle \otimes \langle z_{in}\rangle \\ &= \text{cond}\left(\chi_{\geq}(y^p_p, 2 \cdot y^p_{len}), nis, \langle z_{in}\rangle\right) \end{aligned}$$

and $nis = \text{cond}\left(\chi_{\geq}(y^{+c}_c, \hat{m}), \dots, x_s\right)$,

all x_s remain at $0 = \langle z_{in}\rangle$ until $y^p_p \geq 2 \cdot y^p_{len} = 2 \cdot |X|$ and $y^{+c}_c \geq \hat{m}$. At that point, we get

$$\begin{aligned} f_4^{\text{act}}(\overline{x}, \dots)_s &= x_s + \sum_{z \in Z \setminus Z_{\text{branch}} \setminus \{z_{out}\}} \chi_{=}(x_s, \langle z\rangle) \otimes (\langle s(z)\rangle - x_s) + \sum_{z \in Z_{\text{branch}}} \chi_{=}(x_s, \langle z\rangle) \otimes (\dots) \\ &= 0 + \langle s(z_{in})\rangle - 0 \quad \text{and} \end{aligned}$$

$$\begin{aligned} f_4^{\text{act}}(\overline{v}, \dots)_c &= v_c + \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{=}(x_p, y^p_p) \otimes \left(y^{out}_c - v_c\right) = v_c \\ = f_3^{\text{act}}(\overline{u}, \dots)_c &= u_c - \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z\rangle) \otimes 1 + \sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z\rangle) \otimes 1 = u_c \\ = f_2^{\text{act}}(\overline{x}, \dots)_c &= c_1 + \chi_{=}(y^{prev}_s, \langle z_{in}\rangle) \otimes \\ &\quad \left(x_p - y^{prev}_{len} - c_1 - \chi_{\geq}(2 \cdot y^{prev}_{len}, x_p) \otimes y^{prev}_{len} + \chi_{\geq}(y^{prev}_{len}, x_p) \otimes (2 \cdot y^{prev}_{len})\right) \\ &= c_1 + x_p - |X| - c_1 - \chi_{\geq}(2 \cdot |X|, x_p) \otimes |X| + \chi_{\geq}(|X|, x_p) \otimes (2 \cdot |X|) \\ &= \begin{cases} x_p & \text{if } x_p \leq |X| \\ x_p - 2 \cdot |X| & \text{if } |X| < x_p \leq 2 \cdot |X| \\ x_p - |X| & \text{if } x_p > 2 \cdot |X| \end{cases}. \end{aligned}$$

Therefore, the set of x_c is $C = \{-|X|+1, \dots, |Y|-|X|\}$ where Y is the transformer output and $|C| = |Y| \geq 2 \cdot |X|$. For x_v, we get

$$\begin{aligned}
f_4^{\text{act}}(\overline{v}, \dots)_v &= v_v + \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_=(x_p, y_p^p) \otimes \left(y_v^{out} - v_v\right) = v_v \\
= f_3^{\text{act}}(\overline{u}, \dots)_v &= u_v + \sum_{z \in Z_{\text{comp}}} \chi_=(u_s, \langle z \rangle) \otimes \sum_{i=1}^{n} \chi_=(u_c, i) \otimes \left(p_z(y_v^1, \dots, y_v^m)_i - u_v\right) = u_v \\
= f_2^{\text{act}}(\overline{x}, \dots)_v &= x_v + \chi_=(x_s, \langle z_{in} \rangle) \otimes \text{cond}\left(\chi_\geq(y_{len}^{prev}, x_p), 0, \chi_\geq(2 \cdot y_{len}^{prev}, x_p) \otimes (1 - x_v)\right) \\
&= x_v + \text{cond}\left(\chi_\geq(|X|, x_p), 0, \chi_\geq(2 \cdot |X|, x_p) \otimes (1 - x_v)\right) \\
&= \begin{cases} x_v & \text{if } x_p \leq |X| \\ 1 & \text{if } |X| < x_p \leq 2 \cdot |X| \\ x_v & \text{if } x_p > 2 \cdot |X| \end{cases}.
\end{aligned}$$

So for $X = (X_1, \dots, X_{|X|})$, the first $|X|$ vectors retain the values of X, the next $|X|$ get value 1 and all others remain at 0. The resulting sequence Y fulfills all conditions in the definition of Φ and its value is

$$\Phi(Y) = \big(s(z_{in}), (\dots, 0, \underbrace{1, \dots, 1}_{|X|}.X_1, \dots, X_{|X|}, 0, \dots)\big).$$

Finally, $f_4^{\text{act}}(\overline{x}, \dots)_{hlt} = \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_\geq(y_c^{out}, x_{len}) = 0$.

2. Let $X \in (R^d)^*$ such that $\Phi(X) = (z, (\dots, 0, v_{c_{\min}}, \dots, v_0.\, v_1, \dots, v_{c_{\max}}, 0, \dots))$. Then Lemma B.2 applies, the set of x_c is $C = \{c_{\min}, \dots, c_{\max}\}$, $c_{\min} \leq 1$, $c_{\max} \geq \hat{m}$ and all vectors share $x_s = \langle z \rangle$ as well as $x_{\text{trim}} = 0$. The input layer adds a zero vector to the end of the sequence. At layer 2, we get $f_2^{\text{act}}(\overline{x}, \dots)_s = \langle z \rangle$ for all vectors as before and $f_2^{\text{act}}(\overline{x}, \dots)_v = x_v + \chi_=(x_s, \langle z_{in} \rangle) \otimes (\dots) = x_v$. The attention head using f_{+c}^{att} always attends to the vector with $y_c = c_{\max}$. The one using f_{-c}^{att} attends to the vector with $y_c = c_{\min}$, if $c_{\min} \leq 0$. Otherwise it attends to the last vector, which has $x_c = 0$. Therefore $f_2^{\text{act}}(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{+c}, \overline{y}^{-c}, \overline{y}^{prev}, \overline{y}^p)_c =$

$$\begin{aligned}
& c_1 + \chi_=(y_s^{prev}, \langle z_{in} \rangle) \otimes (\dots) = c_1 \\
= {} & x_c + \chi_=(x_p, y_p^p) \otimes \\
& \left(y_c^{+c} + 1 + \sum_{z \in Z_{\text{right}}} \chi_=(y_s^{prev}, \langle z \rangle) \otimes \left(\chi_\geq(y_p^p - 1, y_p^{-c}) \otimes (y_c^{-c} - 1) - y_c^{+c} - 1\right) \right) \\
= {} & x_c + \chi_=(x_p, y_p^p) \otimes \\
& \left(c_{\max} + 1 + \sum_{z \in Z_{\text{right}}} \chi_=(y_s^{prev}, \langle z \rangle) \otimes \left(\chi_\geq(y_p^p - 1, y_p^{-c}) \otimes (y_c^{-c} - 1) - c_{\max} - 1\right) \right) \\
= {} & \begin{cases} x_c & \text{for all but the last vector} \\ c_{\max} + 1 & \text{if } z \notin Z_{\text{right}} \\ 0 & \text{if } z \in Z_{\text{right}} \text{ and } c_{\min} = 1, \text{ because } y_p^{-c} = y_p^p \\ c_{\min} - 1 & \text{otherwise} \end{cases}
\end{aligned}$$

The resulting set of x_c is $C \cup \{c_{\max} + 1\}$, if $z \notin Z_{\text{right}}$ and $C \cup \{c_{\min} - 1\}$, if $z \in Z_{\text{right}}$. Layer 2 also sets $f_2^{\text{act}}(\overline{x}, \dots)_{fi} = \chi_=(f_2^{\text{act}}(\overline{x}, \dots)_c, i)$ for $i = 1, \dots, \hat{m}$. Because $c_{\min} \leq 1$ and $c_{\max} \geq \hat{m}$, there is exactly one vector with $x_{fi} = G$ for each i.

At layer 3, the $\overline{y}^1, \dots, \overline{y}^{\hat{m}}$ are exactly those vectors, so we get $f_3^{\text{act}}(\overline{x}, \overline{y}^{-c}, \overline{y}^1, \dots, \overline{y}^{\hat{m}}, \overline{y}^f)_{vari} = v_i$ for $i = 1, \dots, \hat{m}$. If $z \in Z_{\text{comp}}$, we get

$$\begin{aligned}
f_3^{\text{act}}(\overline{x}, \dots)_v &= x_v + \sum_{z \in Z_{\text{comp}}} \chi_=(x_s, \langle z \rangle) \otimes \sum_{i=1}^{n} \chi_=(x_c, i) \otimes \left(p_z(y_v^1, \dots, y_v^m)_i - x_v\right) \\
&= x_v + \sum_{i=1}^{n} \chi_=(x_c, i) \otimes \left(p_z(v_1, \dots, v_m)_i - x_v\right) \\
&= p_z(v_1, \dots, v_m)_{x_c},
\end{aligned}$$

which replaces each v_i in μ with $p_z(v_1, \dots, v_m)_i$. Otherwise, $f_3^{\text{act}}(\overline{x}, \dots)_v = x_v$. If $z \in Z_{\text{left}}$, we get

$$f_3^{\text{act}}(\overline{x}, \dots)_c = x_c - \sum_{z \in Z_{\text{left}}} \chi_=(x_s, \langle z \rangle) \otimes 1 + \sum_{z \in Z_{\text{right}}} \chi_=(x_s, \langle z \rangle) \otimes 1 = x_c - 1,$$

which shifts all values of μ one cell to the left. If $z \in Z_{\text{right}}$, $f_3^{\text{act}}(\overline{x}, \dots)_c = x_c + 1$. Otherwise, $f_3^{\text{act}}(\overline{x}, \dots)_c = x_c$. If $z \notin Z_{\text{branch}}$, the final layer computes the next state as before. If $z \in Z_{\text{branch}}$, we get $f_4^{\text{act}}(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{+c}, \dots)_s =$

$$\begin{aligned}
nis &= \text{cond}\left(\chi_\geq(y_c^{+c}, \hat{m}), x_s + \sum_{z \in Z \setminus Z_{\text{branch}} \setminus \{z_{out}\}} \chi_=(x_s, \langle z \rangle) \otimes (\langle s(z) \rangle - x_s)\right. \\
&\quad \left. + \sum_{z \in Z_{\text{branch}}} \chi_=(x_s, \langle z \rangle) \otimes \Big(\text{cond}\left(\chi_\geq(p_z(x_{var1}, \dots, x_{varm}), 0), \langle s_1(z) \rangle, \langle s_0(z) \rangle\right) - x_s \Big), x_s \right) \\
&= x_s + \text{cond}\left(\chi_\geq(p_z(v_1, \dots, v_m), 0), \langle s_1(z) \rangle, \langle s_0(z) \rangle\right) - x_s \\
&= \begin{cases} \langle s_1(z) \rangle & \text{if } p_z(v_1, \dots, v_m) \geq 0 \\ \langle s_0(z) \rangle & \text{if } p_z(v_1, \dots, v_m) < 0 \end{cases}
\end{aligned}$$

All other values remain unchanged. Therefore $\Phi(X) \vdash \Phi(T_{RC}(M)(X))$.

3. Let $X \in (R^d)^*$ such that $\Phi(X) = (z_{out}, (\dots, 0, v_{c_{\min}}, \dots, v_0. v_1, \dots, v_{c_{\max}}, 0, \dots))$. Then, as seen before, Lemma B.2 applies, $x_{\text{trim}} = 0$ everywhere and layers $1, 2, 3$ do not change the values of x_v, x_c and x_{trim} except for the last vector. At layer 2, we get

$$\begin{aligned}
f_2^{\text{act}}(\overline{x}, \dots)_f &= \chi_=(x_v, 0) \otimes \chi_\geq(0, f_2^{\text{act}}(\overline{x}, \dots)_c) \otimes (f_2^{\text{act}}(\overline{x}, \dots)_c + y_p^p) \\
&= \begin{cases} x_c + y_p^p & \text{if } x_v = 0 \text{ and } x_c \leq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \\
f_2^{\text{act}}(\overline{x}, \dots)_{out} &= \text{cond}\left(\chi_=(y_{\text{trim}}^{prev}, 0), \chi_=(f_2^{\text{act}}(\overline{x}, \dots)_c, 1), \chi_=(f_2^{\text{act}}(\overline{x}, \dots)_c, y_c^{prev} + 1)\right) \\
&= \begin{cases} G & \text{if } x_c = 1 \\ 0 & \text{otherwise} \end{cases}.
\end{aligned}$$

Then, at layer 3, because $-x_c \leq -c_{\min} \leq |X| \leq |X| + 1 = y_p^p$ for all $\overline{x}$, the attention head using f_f^{att} always attends to the vector with the maximum x_c among those with $x_v = 0$ and $x_c \leq 0$, if such a vector exists. Otherwise, it attends to the first vector. Therefore,

$$\begin{aligned}
f_3^{\text{act}}(\overline{x}, \dots)_{len} &= x_{len} + \chi_=(x_s, \langle z_{out} \rangle) \otimes \\
&\qquad \Big(\text{cond}\left(\chi_=(y_v^f, 0) \otimes \chi_\geq(0, y_c^f), -y_c^f, \chi_\geq(0, y_c^{-c}) \otimes (1 - y_c^{-c})\right) - x_{len} \Big) \\
&= \text{cond}\left(\chi_=(y_v^f, 0) \otimes \chi_\geq(0, y_c^f), -y_c^f, \chi_\geq(0, c_{\min}) \otimes (1 - c_{\min})\right) \\
&= \begin{cases} -y_c^f & \text{if there is a vector } \overline{y}^f \text{ with } x_v = 0 \text{ and } x_c \leq 0 \\ 1 - c_{\min} & \text{if } c_{\min} \leq 0 \\ 0 & \text{if } c_{\min} = 1 \end{cases}.
\end{aligned}$$

In all cases, we have $f_3^{\text{act}}(\overline{x}, \dots)_{len} = \min\{i \mid i \geq 0, v_{-i} = 0\}$, when defining $v_{c_{\min}-1} = 0$, which is the correct output length. At layer 4, $\overline{y}^{out}$ is the one vector with $x_{out} = G$, which has $x_c = 1$ and

$x_v = v_1$. So for $\overline{w} = f_4^{\text{act}}(\overline{x}, \dots)$, we get

$$
\begin{aligned}
w_s &= nis = x_s = \langle z_{out} \rangle, \\
w_v &= x_v + \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_=(x_p, y_p^p) \otimes \left(y_v^{out} - x_v\right) \\
&= \begin{cases} v_1 & \text{for the last vector} \\ x_v & \text{for the other vectors} \end{cases}, \\
w_c &= x_c + \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_=(x_p, y_p^p) \otimes \left(y_c^{out} - x_c\right) \\
&= \begin{cases} 1 & \text{for the last vector} \\ x_c & \text{for the other vectors} \end{cases}, \\
w_{\text{trim}} &= x_{\text{trim}} + \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_=(x_p, y_p^p) \otimes \left(\text{cond}\left(\chi_\geq(0, x_{len}), x_{\text{trim}}, 1\right) - x_{\text{trim}}\right) \\
&= \begin{cases} G & \text{for the last vector, if } x_{len} > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \\
w_{hlt} &= \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_\geq(y_c^{out}, x_{len}) \\
&= \begin{cases} G & \text{if } x_{len} \leq 1 \\ 0 & \text{otherwise} \end{cases}.
\end{aligned}
$$

Therefore, if the output length $x_{len} = 0$, then $x_{hlt} \neq 0$ and $x_{\text{trim}} = 0$ everywhere. The resulting sequence after trim is empty. If $x_{len} = 1$, then $x_{hlt} \neq 0$, but $x_{\text{trim}} = G$ for the last vector, which has $x_v = v_1$. The resulting sequence after trim is (v_1). So both of these cases have the correct output. If $x_{len} > 1$, then $x_{hlt} = 0$ and $x_{\text{trim}} = G$ for the last vector. Then, in the next iteration, we get

$$
f_2^{\text{act}}(\overline{x}, \dots)_{out} = \chi_=(f_2^{\text{act}}(\overline{x}, \dots)_c, y_c^{prev} + 1) = \chi_=(f_2^{\text{act}}(\overline{x}, \dots)_c, 2),
$$

because $y_{\text{trim}}^{prev} = G$. So the next $\overline{y}^{out}$ in layer 4 has $y_c^{out} = 2$, while the computation of x_{len} is the same. The previous vector retains $w_{\text{trim}} = 1$ and the last vector gets $w_{\text{trim}} = 1$, $w_c = 2$ and $w_v = v_2$. This argument is applied inductively until $w_c \geq x_{len}$ for the last vector. The resulting sequence after trim is $(v_1, \dots, v_{x_{len}})$, which is the correct output of M. □

Definition B.3. Let M be a $\mathbf{BSS}_R$-machine with state set Z like in Definition B.1.

Because the values of the first vector can only change temporarily in **S-𝔠**, we need to change the definition of $\chi_\geq$. Let $\chi'_\geq$ be the definition from Definition B.1. Then define

$$
\chi_\geq(a, b) = \text{cond}\left(\chi'_\geq(\chi'_\geq(0,0) \cdot 2, 1), \chi'_\geq(a, b), \text{cond}\left(\chi'_\geq(b, a), 0, G\right)\right).
$$

The value $\chi'_\geq(0,0)$ needs to be computed only once, but this does require an additional layer before all others.

The first layer uses one attention head with the attention function $f_p^{\text{att}}(\overline{x}, \overline{y}) = x_{one} \cdot y_p$. Its actvation function sets $f_1^{\text{act}}(\overline{x}, \overline{y}^p)_{end} = \chi_\geq(2 \cdot x_p + 1, 2 \cdot y_p^p)$ and $f_1^{\text{act}}(\overline{x}, \overline{y}^p)_n = y_p^p$. All coordinates, which are not described here, are retained from $\overline{x}$.

The second layer uses the attention functions $f^{\text{att}}_{\text{sign}}$, f^{att}_{prev} and f^{att}_{p}.

$$
\begin{aligned}
f^{\text{act}}_2(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{prev}, \overline{y}^{p}) &= \overline{u} \\
lni &= \chi_{=}(x_p, y^p_p) \otimes \text{cond}\left(\chi_{=}(y^{prev}_s, \langle z_{in}\rangle), 0, G\right) \\
u_c &= \text{cond}\left(y^{\text{sign}}_{\text{sign}}, x_c + \chi_{=}(x_p, y^p_p) \otimes \text{cond}\left(\chi_{=}(y^{prev}_s, \langle z_{in}\rangle),\right.\right. \\
&\qquad \left.\left. \text{cond}\left(\chi_{>}(3 \cdot y^{prev}_{len} + 2, x_p), y^{prev}_c + 1, 1\right), y^{prev}_c\right), 1 - y^p_p\right) \\
u_v &= x_v + lni \otimes y^{prev}_v \\
u_{vi} &= x_{vi} + lni \otimes y^{prev}_{vi} \quad \text{for } i = 1, \ldots, \hat{m} - 1 \\
u_s &= x_s + \chi_{=}(x_p, y^p_p) \otimes y^{prev}_s \\
u_{len} &= \text{cond}\left(y^{\text{sign}}_{\text{sign}}, x_{len} + \chi_{=}(x_p, y^p_p) \otimes y^{prev}_{len}, y^p_p - 1\right) \\
u_f &= \chi_{=}(x_p, y^{prev}_c + 1) \\
u_{fi} &= \text{cond}\left(\chi_{=}(y^{prev}_s, \langle z_{in}\rangle), \chi_{=}(x_c, i),\right. \\
&\qquad \text{cond}\left(\chi_{=}(y_s, \langle z_{out}\rangle), \chi_{\geq}(0, y_{len}) \otimes \chi_{>}(y^p_p, x_p), G\right) \otimes \\
&\qquad \left.\chi_{\geq}(y^{prev}_c + i - 1 - u_c, 0) \otimes \chi_{>}(\hat{m}, y^{prev}_c + i - 1 - u_c)\right) \quad \text{for } i = 0, \ldots, \hat{m} + 1
\end{aligned}
$$

The third layer uses the attention functions $f^{\text{att}}_{\text{sign}}$, f^{att}_{f}, $f^{\text{att}}_{f'i}(\overline{x}, \overline{y}) = x_n \cdot y_{fi} + x_{one} \cdot y_p$ for $i = 0, \ldots, \hat{m} + 1$, $f^{\text{att}}_{\text{trim}}(\overline{x}, \overline{y}) = x_{one} \cdot y_{\text{trim}}$ and f^{att}_{p}.

$$
\begin{aligned}
f^{\text{act}}_3(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{in}, \overline{y}^{0}, \ldots, \overline{y}^{\hat{m}+1}, \overline{y}^{\text{trim}}, \overline{y}^{p}) &= \overline{v} \\
v_{vari} &= \chi_{=}(y^i_c, x_c + i - 1) \otimes y^i_v + \sum_{j=1}^{\hat{m}-1} \chi_{=}(x_c + i - 1 - y^i_c, j) \otimes y^i_{vj} \quad \text{for } i = 0, \ldots, \hat{m} + 1 \\
v_v &= x_v + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y^p_p) \otimes \Big(\sum_{z \in Z_{\text{comp}}} \chi_{=}(x_s, \langle z\rangle) \otimes (p_z(v_{var1}, \ldots, v_{varm})_1 - x_v) + \\
&\qquad \sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z\rangle) \otimes (v_{var0} - x_v) + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z\rangle) \otimes (v_{var2} - x_v) + \\
&\qquad \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes (v_{var2} - x_v) + \\
&\qquad \chi_{=}(x_s, \langle z_{in}\rangle) \otimes \Big(\text{cond}\left(\chi_{\geq}(2 \cdot x_{len} + 1, x_p), 1,\right. \\
&\qquad\qquad \left. \text{cond}\left(\chi_{\geq}(3 \cdot x_{len} + 1, x_p), y^{in}_v, v_{var1}\right)\right) - x_v\Big)\Big) \\
v_{vi} &= x_{vi} + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y^p_p) \otimes \Big(\sum_{z \in Z_{\text{comp}} \text{ where } n > i} \chi_{=}(x_s, \langle z\rangle) \otimes (p_z(v_{var1}, \ldots, v_{varm})_{i+1} - x_{vi}) + \\
&\qquad \sum_{z \in Z_{\text{right}}(v_{vari} - x_{vi})} \chi_{=}(x_s, \langle z\rangle) \otimes + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z\rangle) \otimes (v_{vari+2} - x_{vi}) + \\
&\qquad \chi_{=}(x_s, \langle z_{in}\rangle) \otimes \chi_{>}(x_p, 3 \cdot x_{len} + 1) \otimes (v_{var(i+1)} - x_{vi})\Big) \quad \text{for } i = 1, \ldots, \hat{m} - 1
\end{aligned}
$$

$$
\begin{aligned}
ct &= \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes (y^p_p - y^{\text{trim}}_p) \\
co &= \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{=}(ct, 0) \otimes \text{cond}\left(\chi_{=}(v_{var0}, 0), 0, 1\right) \\
v_c &= x_c + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y^p_p) \otimes \Big(\sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z\rangle) \otimes -1 + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z\rangle) \otimes 1 + \\
&\qquad \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes 1 - co\Big) \\
v_{len} &= x_{len} + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y^p_p) \otimes co
\end{aligned}
$$

The fourth layer uses the attention functions $f^{\text{att}}_{\text{sign}}$, f^{att}_{+c}, $f^{\text{att}}_{\text{trim}}$, f^{att}_{prev} and f^{att}_{p}.

$$
\begin{aligned}
f^{\text{act}}_4(\overline{x}, \overline{y}^{\text{sign}}, \overline{y}^{+c}, \overline{y}^{\text{trim}}, \overline{y}^{prev}, \overline{y}^{p}) &= \overline{w} \\
nis &= y^{\text{sign}}_{\text{sign}} \otimes \text{cond}\left(\chi_{=}(x_s, \langle z_{in}\rangle), \chi_{\geq}(y^p_p, 3 \cdot y^p_{len} + 2), G\right) \\
w_{\text{sign}} &= G \\
w_s &= \text{cond}\Big(nis, x_s + \sum_{z \in Z \setminus (Z_{\text{branch}} \cup \{z_{out}\})} \chi_{=}(x_s, \langle z\rangle) \otimes (\langle s(z)\rangle - x_s) + \sum_{z \in Z_{\text{branch}}} \chi_{=}(x_s, \langle z\rangle) \otimes \\
&\qquad \Big(\text{cond}\left(\chi_{\geq}(p_z(x_{var1}, \ldots, x_{varm}), 0), \langle s_1(z)\rangle, \langle s_0(z)\rangle\right) - x_s\Big), x_s\Big) \\
w_{len} &= \text{cond}\left(nis, \sum_{z \in Z \setminus \{z_{out}\}} \chi_{=}(x_s, \langle z\rangle) \otimes (0 - x_{len}) + x_{len}, x_{len}\right) \\
out &= y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{=}(x_{len}, y^{prev}_{len}) \\
ct &= \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes (y^p_p - y^{\text{trim}}_p) \\
w_c &= x_c + out \otimes \chi_{=}(ct, 0) \otimes x_{len} \\
w_{\text{trim}} &= x_{\text{trim}} + out \otimes \Big(\text{cond}\left(\chi_{\geq}(0, x_{len}), x_{\text{trim}}, 1\right) - x_{\text{trim}}\Big) \\
w_{hlt} &= out \otimes \chi_{\geq}(ct + 1, x_{len})
\end{aligned}
$$

Lemma B.4. *Let R be a $\mathbb{Z}$-ordered ring and M be a $\mathbf{BSS}_R$-machine. If X is a sequence of vectors such that the set of x_{sign} in X is $\{1\}$ or $\{0, 1\}$, then for all $a, b \in R$,*

$$\chi_{\geq}(a, b) = \begin{cases} 1 & \text{if } a \geq b \\ 0 & \text{if } a < b \end{cases}$$

within all layers of $T_{SC}(M)(X)$ and the set of x_{sign} in $T_{SC}(M)(X)$ is $\{1\}$ or $\{0, 1\}$.

Proof. After the input layer, the last vector always has $x_{\text{sign}} = 0$, so there is at least one vector with $x_{\text{sign}} = 0$. An attention head computing $\chi'_{\geq}(a, b)$ for $\overline{x}$ uses $f^{\text{att}}(\overline{x}, \overline{y}) = x_{\geq} \cdot y_{\text{sign}}$ with $x_{\geq} = a - b$. Therefore,

- if $x_{\geq} > 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\text{sign}} = 1$,
- if $x_{\geq} < 0$, $\overline{x}$ attends to a vector $\overline{y}$ with $y_{\text{sign}} = 0$ and
- if $x_{\geq} = 0$, all attention scores are 0, so $\overline{x}$ attends to the first vector.

If the first vector has $x_{\text{sign}} = 1$, then $\chi'_{\geq}(0, 0) \cdot 2 = 2 > 1$ and

$$\text{cond}\left(\chi'_{\geq}(\chi'_{\geq}(0, 0) \cdot 2, 1), \chi'_{\geq}(a, b), \text{cond}\left(\chi'_{\geq}(b, a), 0, G\right)\right) = \chi'_{\geq}(a, b) = \begin{cases} 1 & \text{if } a \geq b \\ 0 & \text{if } a < b \end{cases}.$$

If the first vector has $x_{\text{sign}} = 0$, then $\chi'_{\geq}(0, 0) \cdot 2 = 0 < 1$ and

$$\text{cond}\left(\chi'_{\geq}(\chi'_{\geq}(0, 0) \cdot 2, 1), \chi'_{\geq}(a, b), \text{cond}\left(\chi'_{\geq}(b, a), 0, G\right)\right) = \text{cond}\left(\chi'_{\geq}(b, a), 0, 1\right) = \begin{cases} 0 & \text{if } b > a \\ 1 & \text{if } b \leq a \end{cases}.$$

Therefore the computation of $\chi_{\geq}$ always produces the correct output. As x_{sign} cannot change until the final layer, this is the case for all layers. In the final layer, we get $f^{\text{act}}_4(x, \ldots)_{\text{sign}} = G$ for all vectors, but only the last vector is part of the output sequence. The resulting set of x_{sign} is the same as for X. □

Theorem 4.2. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{S\text{-}FT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\text{left})\big)$ *contains all $\mathbf{BSS}_R$-computable functions (without division).*

Proof. This proof has the same structure as the proof of Theorem 4.1, but uses T_{SC} instead of T_{RC}. We define a different function Φ and proof the same three statements as before.

For any vector sequence $Y = (\overline{y}^1, \dots, \overline{y}^{|Y|})$, if $y_s^{|Y|} = \langle z \rangle$ for some $z \in Z$,

$$\{y_c^i, y_c^i + 1, \dots, y_c^i + \hat{m} - 1 \mid 1 \le i \le |Y|\} = \{c_{\min}, c_{\min} + 1, \dots, c_{\max} + \hat{m} - 1\},$$

the set of y_{sign} in Y is $\{1\}$ or $\{0, 1\}$, $y_{len}^{|Y|} = 0$ and $y_{\text{trim}} = 0$ everywhere, then

$$\Phi(Y) = (z, (\dots, 0, v_{c_{\min}}, \dots, v_{(y_c^{|Y|}-1)}. v_{y_c^{|Y|}}, \dots, v_{c_{\max}+\hat{m}-1}, 0, \dots))$$

where

$$v_j = \begin{cases} w_v^j & \text{if } w_c^j = j \\ w^j_{v(j-w_c^j)} & \text{if } w_c^j < j < w_c^j + \hat{m} \end{cases} \quad \text{and} \quad \overline{w}^j = \overline{y}^k \text{ for } k = \max\{i \mid y_c^i \le i < y_c^i + \hat{m}\}.$$

1. During the first iteration, in the same way as before, we get $f_2^{\text{act}}(\overline{x}, \dots)_{len} = |X|$ and x_{sign} remains zero until layer 4 for all vectors. However, instead of $\chi_{\ge}(a, b) = 0$ for all a, b, we get

$$\begin{aligned} \chi_{\ge}(a,b) &= \text{cond}\left(\chi'_{\ge}(\chi'_{\ge}(0,0)\cdot 2, 1), \chi'_{\ge}(a,b), \text{cond}\left(\chi'_{\ge}(b,a), 0, G\right)\right) \\ &= \text{cond}\left(0, 0, \text{cond}\left(0, 0, G\right)\right) \\ &= G. \end{aligned}$$

Therefore, the last vector in the sequence gets

$$\begin{aligned} lni &= \chi_{=}(x_p, y_p^p) \otimes \text{cond}\left(\chi_{=}(y_s^{prev}, \langle z_{in} \rangle), 0, G\right) = G \otimes \text{cond}\left(G, 0, G\right) = 0 \\ f_2^{\text{act}}(\overline{x}, \dots)_c &= \text{cond}\left(y_{\text{sign}}^{\text{sign}}, \dots, 1 - y_p^p\right) = 1 - y_p^p = -|X| \\ f_2^{\text{act}}(\overline{x}, \dots)_v &= x_v + lni \otimes y_v^{prev} = x_v = 0 \\ f_2^{\text{act}}(\overline{x}, \dots)_{vi} &= x_{vi} + lni \otimes y_{vi}^{prev} = x_{vi} = 0 \quad \text{for } i = 1, \dots, \hat{m} - 1 \\ f_2^{\text{act}}(\overline{x}, \dots)_s &= x_s + \chi_{=}(x_p, y_p^p) \otimes y_s^{prev} = 0 + 0 = \langle z_{in} \rangle \end{aligned}$$

$$\begin{aligned} f_3^{\text{act}}(\overline{x}, \dots)_c &= x_c + y_{\text{sign}}^{\text{sign}} \otimes (\dots) = x_c = -|X| \\ f_3^{\text{act}}(\overline{x}, \dots)_v &= x_v + y_{\text{sign}}^{\text{sign}} \otimes (\dots) = x_v = 0 \\ f_3^{\text{act}}(\overline{x}, \dots)_{vi} &= x_{vi} + y_{\text{sign}}^{\text{sign}} \otimes (\dots) = x_{vi} = 0 \quad \text{for } i = 1, \dots, \hat{m} - 1 \\ f_3^{\text{act}}(\overline{x}, \dots)_{len} &= x_{len} + y_{\text{sign}}^{\text{sign}} \otimes (\dots) = x_{len} = |X| \end{aligned}$$

$$\begin{aligned} nis &= y_{\text{sign}}^{\text{sign}} \otimes (\dots) = 0 \\ f_4^{\text{act}}(\overline{x}, \dots)_s &= \text{cond}\left(nis, \dots, x_s\right) = x_s = \langle z_{in} \rangle \\ out &= y_{\text{sign}}^{\text{sign}} \otimes (\dots) = 0 \\ f_4^{\text{act}}(\overline{x}, \dots)_c &= x_c + out \otimes (\dots) = x_c = -|X| \\ f_4^{\text{act}}(\overline{x}, \dots)_{len} &= \text{cond}\left(nis, \dots, x_{len}\right) = x_{len} = |X| \\ f_4^{\text{act}}(\overline{x}, \dots)_{\text{sign}} &= G \\ f_4^{\text{act}}(\overline{x}, \dots)_{\text{trim}} &= x_{\text{trim}} + out \otimes (\dots) = x_{\text{trim}} = 0 \quad \text{and} \\ f_4^{\text{act}}(\overline{x}, \dots)_{hlt} &= out \otimes (\dots) = 0. \end{aligned}$$

Therefore, Lemma B.4 applies to all following iterations. In the next iteration, $\overline{y}^{prev}$ is always the second-to-last vector and for the last vector, we get

$$\begin{aligned}
f_2^{\text{act}}(\overline{x},\dots)_s &= x_s + \chi_{=}(x_p, y_p^p) \otimes y_s^{prev} = 0 + \chi_{=}(x_p, y_p^p) \otimes 0 = 0 = \langle z_{in} \rangle \\
f_2^{\text{act}}(\overline{x},\dots)_{len} &= \text{cond}\left(y_{\text{sign}}^{\text{sign}}, x_{len} + \chi_{=}(x_p, y_p^p) \otimes y_{len}^{prev}, y_p^p - 1\right) \\
&= 0 + \chi_{=}(x_p, y_p^p) \otimes |X| = |X|
\end{aligned}$$

$$\begin{aligned}
co &= \chi_{=}(x_s, \langle z_{out} \rangle) \otimes (\dots) = 0 \\
f_3^{\text{act}}(\overline{x},\dots)_{len} &= x_{len} + y_{\text{sign}}^{\text{sign}} \otimes co \otimes (\dots) = x_{len} \\
nis &= y_{\text{sign}}^{\text{sign}} \otimes \text{cond}\left(\chi_{=}(x_s, \langle z_{in} \rangle), \chi_{\geq}(y_p^p, 3 \cdot y_{len}^p + 2), G\right) \\
&= G \otimes \text{cond}\left(G, \chi_{\geq}(y_p^p, 3 \cdot y_{len}^p + 2), G\right) \\
&= \chi_{\geq}(y_p^p, 3 \cdot y_{len}^p + 2)
\end{aligned}$$

$$\begin{aligned}
f_4^{\text{act}}(\overline{x},\dots)_s &= \text{cond}\left(nis, \dots, x_s\right) \quad \text{and} \\
f_4^{\text{act}}(\overline{x},\dots)_{len} &= \text{cond}\left(nis, \dots, x_{len}\right).
\end{aligned}$$

So x_s remains zero until $y_p^p \geq 3 \cdot y_{len}^p + 2 = 3 \cdot |X| + 2$. For the other values of the last vector, we get

$$\begin{aligned}
lni &= \chi_{=}(x_p, y_p^p) \otimes \text{cond}\left(\chi_{=}(y_s^{prev}, \langle z_{in} \rangle), 0, G\right) = 0 \\
f_2^{\text{act}}(\overline{x},\dots)_c &= \text{cond}\left(y_{\text{sign}}^{\text{sign}}, x_c + \chi_{=}(x_p, y_p^p) \otimes \text{cond}\left(\chi_{=}(y_s^{prev}, \langle z_{in} \rangle),\right.\right. \\
&\qquad \left.\left. \text{cond}\left(\chi_{>}(3 \cdot y_{len}^{prev} + 2, x_p), y_c^{prev} + 1, 1\right), y_c^{prev}\right), 1 - y_p^p\right) \\
&= 0 + \text{cond}\left(\chi_{>}(3 \cdot |X| + 2, x_p), y_c^{prev} + 1, 1\right) \\
&= \begin{cases} y_c^{prev} + 1 & \text{if } x_p < 3 \cdot |X| + 2 \\ 1 & \text{otherwise} \end{cases} \\
f_2^{\text{act}}(\overline{x},\dots)_v &= x_v + lni \otimes y_v^{prev} = x_v = 0 \\
f_2^{\text{act}}(\overline{x},\dots)_{vi} &= x_{vi} + lni \otimes y_{vi}^{prev} = x_{vi} = 0 \quad \text{for } i = 1, \dots, \hat{m} - 1 \\
f_3^{\text{act}}(\overline{x},\dots)_c &= x_c + y_{\text{sign}}^{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z \rangle) \otimes -1 + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z \rangle) \otimes 1 + \\
&\qquad \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{>}(y_{\text{trim}}^{\text{trim}}, 0) \otimes 1 - co \\
&= x_c - co = x_c \\
f_3^{\text{act}}(\overline{x},\dots)_v &= x_v + y_{\text{sign}}^{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \Big(\sum_{z \in Z_{\text{comp}}} \chi_{=}(x_s, \langle z \rangle) \otimes (p_z(v_{var1}, \dots, v_{varm})_1 - x_v) \\
&\qquad + \sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z \rangle) \otimes (v_{var0} - x_v) + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z \rangle) \otimes (v_{var2} - x_v) + \\
&\qquad \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{>}(y_{\text{trim}}^{\text{trim}}, 0) \otimes (v_{var2} - x_v) + \\
&\qquad \chi_{=}(x_s, \langle z_{in} \rangle) \otimes \Big(\text{cond}\left(\chi_{\geq}(2 \cdot x_{len} + 1, x_p), 1,\right. \\
&\qquad\qquad \left. \text{cond}\left(\chi_{\geq}(3 \cdot x_{len} + 1, x_p), y_v^{in}, v_{var1}\right)\right) - x_v \Big) \Big) \\
&= x_v + \text{cond}\left(\chi_{\geq}(2 \cdot |X| + 1, x_p), 1,\right. \\
&\qquad \left. \text{cond}\left(\chi_{\geq}(3 \cdot |X| + 1, x_p), y_v^{in}, v_{var1}\right)\right) - x_v \\
&= \begin{cases} 1 & \text{if } x_p \leq 2 \cdot |X| + 1 \\ y_v^{in} & \text{if } x_p \leq 3 \cdot |X| + 1 \\ v_{var1} & \text{otherwise} \end{cases}
\end{aligned}$$

$$
\begin{aligned}
f_3^{\text{act}}(\overline{x},\dots)_{vi} &= x_{vi} + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \Big(\cdots + \\
&\qquad \chi_{=}(x_s, \langle z_{in}\rangle) \otimes \chi_{>}(x_p, 3 \cdot x_{len} + 1) \otimes (v_{var(i+1)} - x_{vi})\Big) \\
&= \begin{cases} v_{var(i+1)} & \text{if } x_p > 3 \cdot |X| + 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, \dots, \hat{m} - 1 \\
out &= y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{=}(x_{len}, y^{prev}_{len}) = 0 \\
f_4^{\text{act}}(\overline{x},\dots)_c &= x_c + out \otimes (\dots) = x_c \\
f_4^{\text{act}}(\overline{x},\dots)_{\text{trim}} &= x_{\text{trim}} + out \otimes (\dots) = x_{\text{trim}} = 0 \quad \text{and} \\
f_4^{\text{act}}(\overline{x},\dots)_{hlt} &= out \otimes (\dots) = 0.
\end{aligned}
$$

Therefore, x_c increases by 1 in each iteration until $x_p = 3 \cdot |X| + 2$. Then it is set to 1. The first $|X|$ vectors get $x_v = 1$ and the next $|X|$ vectors get $x_v = y_v^{in}$. The attention head computing $\overline{y}^{in}$ uses f_f^{att} and there is always exactly one vector with $f_2^{\text{act}}(\overline{x},\dots)_f = \chi_{=}(x_p, y_c^{prev} + 1) = G$. Since the $(|X| + 1)$-th vector gets $x_c = -|X| + |X| + 1 = 1$, it copies the value from the first vector in the sequence, which is X_1. This is repeated until all input values have been copied to vectors with $x_c = 1, \dots, |X|$. The x_{vi} are all zero except for the last vector.

That vector, which has $x_p = 3 \cdot |X| + 2$ and $x_c = 1$, uses the values from the attention heads using the $f_{f'i}^{\text{att}}$. For $i = 1, \dots, \hat{m}$, there is at most one vector with

$$
f_2^{\text{act}}(\overline{x},\dots)_{fi} = \text{cond}\left(\chi_{=}(y_s^{prev}, \langle z_{in}\rangle), \chi_{=}(x_c, i), \dots\right) = G
$$

and we have shown that this vector contains $x_v = X_i$. If $i > |X|$, such a vector does not exist and the corresponding attention head attends to the first vector. Therefore, for the last vector at layer 3, if $1 \le i \le |X|$, we get $y_c^i = i$, $y_v^i = X_i$ and

$$
\begin{aligned}
v_{vari} &= \chi_{=}(y_c^i, x_c + i - 1) \otimes y_v^i + \sum_{j=1}^{\hat{m}-1} \chi_{=}(x_c + i - 2 - y_c^i, j) \otimes y_{vj}^i \\
&= \chi_{=}(i, 1 + i - 1) \otimes X_i + \sum_{j=1}^{\hat{m}-1} \chi_{=}(1 + i - 2 - i, j) \otimes y_{vj}^i \\
&= X_i + 0.
\end{aligned}
$$

If $i > |X|$, we get

$$
\begin{aligned}
v_{vari} &= \chi_{=}(y_c^i, x_c + i - 1) \otimes y_v^i + \sum_{j=1}^{\hat{m}-1} \chi_{=}(x_c + i - 2 - y_c^i, j) \otimes y_{vj}^i \\
&= \chi_{=}(0, 1 + i - 1) \otimes y_v^i + \sum_{j=1}^{\hat{m}-1} \chi_{=}(1 + i - 2 - 0, j) \otimes 0 = 0 + 0,
\end{aligned}
$$

$$
\text{since } y_c^i = \begin{cases} 0 & \text{if } |X| > 0 \\ -|X| = 0 & \text{if } |X| = 0 \end{cases} \text{ and } y_{vj}^i = 0 \text{ for all } j.
$$

Finally, at layer 4, for the last vector we get

$$
\begin{aligned}
f_4^{\text{act}}(\overline{x},\dots)_s &= \text{cond}\left(nis, x_s + \sum_{z \in Z \setminus (Z_{\text{branch}} \cup \{z_{out}\})} \chi_{=}(x_s, \langle z\rangle) \otimes (\langle s(z)\rangle - x_s) + \sum_{z \in Z_{\text{branch}}} \chi_{=}(x_s, \langle z\rangle) \otimes \right. \\
&\qquad \left. \Big(\text{cond}\left(\chi_{\ge}(p_z(x_{var1}, \dots, x_{varm}), 0), \langle s_1(z)\rangle, \langle s_0(z)\rangle\right) - x_s\Big), x_s\right) \\
&= \text{cond}\left(G, x_s + \chi_{=}(x_s, \langle z_{in}\rangle) \otimes (\langle s(z_{in})\rangle - x_s) + 0, x_s\right) = \langle s(z_{in})\rangle \quad \text{and} \\
f_4^{\text{act}}(\overline{x},\dots)_{len} &= \text{cond}\left(nis, \sum_{z \in Z \setminus \{z_{out}\}} \chi_{=}(x_s, \langle z\rangle) \otimes (0 - x_{len}) + x_{len}, x_{len}\right) \\
&= \sum_{z \in Z \setminus \{z_{out}\}} \chi_{=}(x_s, \langle z\rangle) \otimes (0 - x_{len}) + x_{len} = 0
\end{aligned}
$$

The output sequence Y from this iteration fulfills all conditions for $\Phi(Y) = (s(z_{in}), \mathcal{I}_\infty(X))$.

2. Let $X \in (R^d)^*$ such that

$$\Phi(X) = (z, (\dots, 0, v_{c_{\min}}, \dots, v_{c_{\text{head}}-1}. v_{c_{\text{head}}}, \dots, v_{c_{\max}+\hat{m}-1}, 0, \dots)).$$

Then Lemma B.4 applies and for the last vector at layer 2, we get

$$\begin{aligned}
lni &= \chi_{=}(x_p, y_p^p) \otimes \text{cond}\left(\chi_{=}(y_s^{prev}, \langle z_{in}\rangle), 0, G\right) = G \\
f_2^{\text{act}}(\overline{x}, \dots)_c &= \text{cond}\left(y_{\text{sign}}^{\text{sign}}, x_c + \chi_{=}(x_p, y_p^p) \otimes \text{cond}\left(\chi_{=}(y_s^{prev}, \langle z_{in}\rangle), \dots, y_c^{prev}\right), 1 - y_p^p\right) \\
&= y_c^{prev} = c_{\text{head}} \\
f_2^{\text{act}}(\overline{x}, \dots)_v &= x_v + lni \otimes y_v^{prev} = y_v^{prev} = v_{c_{\text{head}}} \\
f_2^{\text{act}}(\overline{x}, \dots)_{vi} &= x_{vi} + lni \otimes y_{vi}^{prev} = y_{vi}^{prev} = v_{c_{\text{head}}+i} \quad \text{for } i = 1, \dots, \hat{m}-1 \text{ and} \\
f_2^{\text{act}}(\overline{x}, \dots)_s &= x_s + \chi_{=}(x_p, y_p^p) \otimes y_s^{prev} = y_s^{prev} = \langle z\rangle.
\end{aligned}$$

These values remain unchanged for all other vectors. For $i = 0, \dots, \hat{m}+1$ and for each vector, we get

$$\begin{aligned}
f_2^{\text{act}}(\overline{x}, \dots)_{fi} = \text{cond}\big(&\chi_{=}(y_s^{prev}, \langle z_{in}\rangle), \chi_{=}(x_c, i), \\
&\text{cond}\left(\chi_{=}(y_s, \langle z_{out}\rangle), \chi_{\geq}(0, y_{len}) \otimes \chi_{>}(y_p^p, x_p), G\right) \otimes \\
&\chi_{\geq}(y_c^{prev} + i - 1 - f_2^{\text{act}}(\overline{x}, \dots)_c, 0) \otimes \chi_{>}(\hat{m}, y_c^{prev} + i - 1 - f_2^{\text{act}}(\overline{x}, \dots)_c)\big) \\
= \chi_{\geq}(c_{\text{head}} + i - 1 - f_2^{\text{act}}(\overline{x}, \dots)_c, 0) &\otimes \chi_{>}(\hat{m}, c_{\text{head}} + i - 1 - f_2^{\text{act}}(\overline{x}, \dots)_c).
\end{aligned}$$

So, for the last vector, $x_{fi} = G$ if $1 \leq i \leq \hat{m}$ and zero otherwise. For any $c_{\min} \leq i < c_{\max} + \hat{m} - c_{\text{head}}$, there must be at least one vector with $x_{fi} = G$. An attention head for $\overline{y}^i$ in layer 3 uses $f_{f'i}^{\text{att}}$, so it always attends to the last such vector in the sequence, if it exists. Due to the definition of the v_j, that vector must have $x_v = v_{c_{\text{head}}+i-1}$ if $x_c = c_{\text{head}} + i - 1$ or $x_{vj} = v_{c_{\text{head}}+i-1}$ for $j = c_{\text{head}} + i - 1 - x_c$. Then, for the last vector, we get $f_3^{\text{act}}(\overline{x}, \dots)_{vari} = v_{vari}$

$$\begin{aligned}
&\chi_{=}(y_c^i, x_c + i - 1) \otimes y_v^i + \sum_{j=1}^{\hat{m}-1} \chi_{=}(x_c + i - 1 - y_c^i, j) \otimes y_{vj}^i \\
&= \chi_{=}(y_c^i, c_{\text{head}} + i - 1) \otimes y_v^i + \sum_{j=1}^{\hat{m}-1} \chi_{=}(c_{\text{head}} + i - 1 - y_c^i, j) \otimes y_{vj}^i \\
&= \begin{cases}
y_v^i = v_{c_{\text{head}}+i-1} & \text{if } y_c^i = c_{\text{head}} + i - 1 \\
y_{vj}^i = v_{c_{\text{head}}+i-1} \text{ with } j = c_{\text{head}} + i - 1 - y_c^i & \text{if } y_c^i \leq c_{\text{head}} + i - 1 < y_c^i + \hat{m} - 1 \\
0 & \text{otherwise}
\end{cases}.
\end{aligned}$$

Therefore, the x_{vari} always contain the correct tape values around the head. If $z \in Z_{\text{comp}}$, we get

$$\begin{aligned}
f_3^{\text{act}}(\overline{x}, \dots)_v &= x_v + y_{\text{sign}}^{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \\
&\qquad \Big(\sum_{z \in Z_{\text{comp}}} \chi_{=}(x_s, \langle z\rangle) \otimes (p_z(v_{var1}, \dots, v_{varm})_1 - x_v) + \dots\Big) \\
&= p_z(v_{var1}, \dots, v_{varm})_1 \\
f_3^{\text{act}}(\overline{x}, \dots)_{vi} &= x_{vi} + y_{\text{sign}}^{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \\
&\qquad \Big(\sum_{z \in Z_{\text{comp}} \text{ where } n > i} \chi_{=}(x_s, \langle z\rangle) \otimes (p_z(v_{var1}, \dots, v_{varm})_{i+1} - x_{vi}) + \dots\Big) \\
&= \begin{cases}
p_z(v_{var1}, \dots, v_{varm})_{i+1} & \text{if } p_z \text{ outputs at least } i+1 \text{ values} \\
x_{vari} & \text{otherwise}
\end{cases} \quad \text{and} \\
f_3^{\text{act}}(\overline{x}, \dots)_c &= x_c + y_{\text{sign}}^{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \Big(\sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z\rangle) \otimes -1 + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z\rangle) \otimes 1 + \\
&\qquad \chi_{=}(x_s, \langle z_{out}\rangle) \otimes \chi_{>}(y_{\text{trim}}^{\text{trim}}, 0) \otimes 1 - co\Big) \\
&= x_c.
\end{aligned}$$

So $x_v, x_{var1}, \ldots, x_{var(\hat{m}-1)}$ contain the output of p_z. If $z \in Z_{\text{left}}$, we get

$$\begin{aligned}
f_3^{\text{act}}(\overline{x}, \ldots)_v &= x_v + y_{\text{sign}}^{\text{sign}} \otimes \chi_=(x_p, y_p^p) \otimes \Big(\cdots + \sum_{z \in Z_{\text{left}}} \chi_=(x_s, \langle z \rangle) \otimes (v_{var2} - x_v) + \ldots \Big) \\
&= v_{var2} \\
f_3^{\text{act}}(\overline{x}, \ldots)_{vi} &= x_{vi} + y_{\text{sign}}^{\text{sign}} \otimes \chi_=(x_p, y_p^p) \otimes \Big(\sum_{z \in Z_{\text{left}}} \chi_=(x_s, \langle z \rangle) \otimes (v_{vari+2} - x_{vi}) + \ldots \Big) \quad \text{and} \\
f_3^{\text{act}}(\overline{x}, \ldots)_c &= x_c + y_{\text{sign}}^{\text{sign}} \otimes \chi_=(x_p, y_p^p) \otimes \Big(\sum_{z \in Z_{\text{right}}} \chi_=(x_s, \langle z \rangle) \otimes -1 + \sum_{z \in Z_{\text{left}}} \chi_=(x_s, \langle z \rangle) \otimes 1 + \ldots \Big) \\
&= x_c + 1.
\end{aligned}$$

So $x_v, x_{var1}, \ldots, x_{var(\hat{m}-2)}$ are shifted to the left, $x_{var(\hat{m}-1)}$ gets the next value from the tape and x_c increases by 1. For $z \in Z_{\text{right}}$, the shift happens to the right instead. If $z \in Z_{\text{branch}}$, we get $f_3^{\text{act}}(\overline{x}, \ldots)_v = x_v$, $f_3^{\text{act}}(\overline{x}, \ldots)_{vi} = x_{vi}$ and $f_3^{\text{act}}(\overline{x}, \ldots)_c = x_c$.

At layer 4, we then get

$$\begin{aligned}
w_s &= \text{cond}\Big(nis, x_s + \sum_{z \in Z \setminus (Z_{\text{branch}} \cup \{z_{out}\})} \chi_=(x_s, \langle z \rangle) \otimes (\langle s(z) \rangle - x_s) + \sum_{z \in Z_{\text{branch}}} \chi_=(x_s, \langle z \rangle) \otimes \\
&\qquad \Big(\text{cond}\big(\chi_{\geq}(p_z(x_{var1}, \ldots, x_{varm}), 0), \langle s_1(z) \rangle, \langle s_0(z) \rangle\big) - x_s \Big), x_s \Big) \\
&= \begin{cases} \langle s(z) \rangle & \text{if } z \notin Z_{\text{branch}} \\ \langle s_1(z) \rangle & \text{if } z \in Z_{\text{branch}} \text{ and } p_z(x_{var1}, \ldots, x_{varm}) \geq 0 \\ \langle s_0(z) \rangle & \text{if } z \in Z_{\text{branch}} \text{ and } p_z(x_{var1}, \ldots, x_{varm}) < 0 \end{cases} \\
out &= y_{\text{sign}}^{\text{sign}} \otimes \chi_=(x_s, \langle z_{out} \rangle) \otimes \chi_=(x_{len}, y_{len}^{prev}) = 0 \quad \text{and} \\
w_c &= x_c + out \otimes \chi_=(ct, 0) \otimes x_{len} = x_c.
\end{aligned}$$

As before, we also have $x_{len} = 0$ for the last vector and $x_{\text{trim}} = x_{hlt} = 0$ everywhere, so $\Phi(X) \vdash \Phi(T_{SC}(M)(X))$.

3. Let $X \in (R^d)^*$ such that

$$\Phi(X) = (z_{out}, (\ldots, 0, v_{c_{\min}}, \ldots, v_{c_{\text{head}}-1}. v_{c_{\text{head}}}, \ldots, v_{c_{\max}+\hat{m}-1}, 0, \ldots)).$$

Then Lemma B.4 applies, $x_{len} = 0$ for the last vector and layer 2 behaves exaclty like for the other non-input states except for

$$\begin{aligned}
f_2^{\text{act}}(\overline{x}, \ldots)_{fi} &= \text{cond}\big(\chi_=(y_s^{prev}, \langle z_{in} \rangle), \chi_=(x_c, i), \\
&\qquad \text{cond}\big(\chi_=(y_s, \langle z_{out} \rangle), \chi_{\geq}(0, y_{len}) \otimes \chi_>(y_p^p, x_p), G\big) \otimes (\ldots)\big) \\
&= \chi_{\geq}(0, y_{len}) \otimes \chi_>(y_p^p, x_p) \otimes (\ldots) \quad \text{for } i = 0, \ldots, \hat{m}+1.
\end{aligned}$$

Since $(\dots)$ contains the previously described values, this results in setting all $x_{fi} = 0$, if $x_{len} > 0$ or $x_p \geq y_p^p$. At layer 3, for the last vector, we get

$$\begin{aligned}
v_v &= x_v + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \Big(\cdots + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z \rangle) \otimes (v_{var2} - x_v) + \\
&\qquad \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes (v_{var2} - x_v) + \dots \Big) \\
&= \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes (v_{var2} - x_v) \\
ct &= \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes (y_p^p - y_p^{\text{trim}}) \\
&= \begin{cases} y_p^p - y_p^{\text{trim}} & \text{if there is a vector } \overline{y}^{\text{trim}} \text{ with } x_{\text{trim}} > 0 \\ 0 & \text{otherwise} \end{cases} \\
co &= \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{=}(ct, 0) \otimes \text{cond}\left(\chi_{=}(v_{var0}, 0), 0, 1\right) \\
&= \begin{cases} 1 & \text{if } ct = 0 \text{ and } v_{var0} \neq 0 \\ 0 & \text{otherwise} \end{cases} \\
v_c &= x_c + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes \Big(\sum_{z \in Z_{\text{right}}} \chi_{=}(x_s, \langle z \rangle) \otimes -1 + \sum_{z \in Z_{\text{left}}} \chi_{=}(x_s, \langle z \rangle) \otimes 1 + \\
&\qquad \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes 1 - co \Big) \\
&= x_c + \chi_{>}(y^{\text{trim}}_{\text{trim}}, 0) \otimes 1 - co \\
v_{len} &= x_{len} + y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_p, y_p^p) \otimes co = x_{len} + co
\end{aligned}$$

Since $\overline{y}^{\text{trim}}$ is computed using $f^{\text{att}}_{\text{trim}}$, if there is no vector with $x_{\text{trim}} > 0$, x_v does not change. If additionally $x_{var0} \neq 0$, then x_c is decremented and x_{len} is incremented. If there is a vector with $x_{\text{trim}} > 0$, then layer 3 behaves exactly like for a shift left state instead. At layer 4, for the last vector, we get

$$\begin{aligned}
w_{len} &= \text{cond}\left(nis, \sum_{z \in Z \setminus \{z_{out}\}} \chi_{=}(x_s, \langle z \rangle) \otimes (0 - x_{len}) + x_{len}, x_{len}\right) = x_{len} \\
out &= y^{\text{sign}}_{\text{sign}} \otimes \chi_{=}(x_s, \langle z_{out} \rangle) \otimes \chi_{=}(x_{len}, y_{len}^{prev}) = \chi_{=}(x_{len}, y_{len}^{prev}) \\
w_c &= x_c + out \otimes \chi_{=}(ct, 0) \otimes x_{len} \\
&= \begin{cases} x_c + x_{len} & \text{if } x_{len} = y_{len}^{prev} \text{ and } ct = 0 \\ x_c & \text{otherwise} \end{cases} \\
w_{\text{trim}} &= x_{\text{trim}} + out \otimes \Big(\text{cond}\left(\chi_{\geq}(0, x_{len}), x_{\text{trim}}, 1\right) - x_{\text{trim}} \Big) \\
&= \begin{cases} 1 & \text{if } x_{len} = y_{len}^{prev} \text{ and } x_{len} > 0 \\ x_{\text{trim}} & \text{otherwise} \end{cases} \\
w_{hlt} &= out \otimes \chi_{\geq}(ct + 1, x_{len}) \\
&= \begin{cases} G & \text{if } x_{len} = y_{len}^{prev} \text{ and } ct + 1 \geq x_{len} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

Overall, $T_{SC}(M)(X)$ behaves in the following way. Initially, for the last vector $x_{len} = 0$ and $x_c = c_{\text{head}}$. Then, x_c is decremented to inspect the cells to the left of the head until one with value 0 is found. Meanwhile, x_{len} is incremented to count the number of steps taken. Because x_{len} is increasing, layer 4 does not change any values.

When a 0 is found, x_c and x_{len} remain unchanged in layer 3. At that point, x_{len} is the desired output length. If it is zero, we get $f^{\text{act}}_4(\overline{x}, \dots)_{\text{trim}} = 0$ and $f^{\text{act}}_4(\overline{x}, \dots)_{hlt} = G$, so the output is the empty sequence. If $x_{len} = 1$, we get $f^{\text{act}}_4(\overline{x}, \dots)_{\text{trim}} = 1$ and $f^{\text{act}}_4(\overline{x}, \dots)_{hlt} = G$. Since x_v was always copied from the previous vector up to this point, the output is the sequence $(v_{c_{\text{head}}})$. If $x_{len} > 1$, we get $f^{\text{act}}_4(\overline{x}, \dots)_{\text{trim}} = 1$ and $f^{\text{act}}_4(\overline{x}, \dots)_{hlt} = 0$. Additionally,

$$f^{\text{act}}_4(\overline{x}, \dots)_c = x_c + x_{len} = c_{\text{head}} - |X| + |X| = c_{\text{head}}.$$

In the following iterations, the transformer simulates left shift states until $ct + 1 \geq x_{len}$ and sets $f_4^{\text{act}}(\overline{x}, \dots)_{\text{trim}} = 1$ for all new vectors. Because ct is the number of vectors added since the first one with $x_{\text{trim}} = 1$, this results in a sequence with x_{len} such vectors. During the shift operations, all of the added vectors are ignored in the attention for the $f_{f'i}^{\text{att}}$, because they have $x_{len} > 0$, so their flags were set to zero in layer 2. Due to the left shift, the resulting values are those to the right of the tape head. □

C Proofs of Section 5

Lemma C.1. *Let R be an ordered ring and $f \in \mathbf{FF}_R$. Then $\|f(\overline{x})\| \in \mathcal{O}(\|\overline{x}\|)$.*

Proof. For any affine transformation $g\colon R^m \to R^n, \overline{x} \mapsto Ax + \overline{b}$ with $A = (a_{ij})$, we have

$$\begin{aligned}
\|g(\overline{x})\| &= \|Ax + \overline{b}\| \\
&= \max\{|\sum_{i=1}^{m} a_{ij} \times x_i + b_i| \quad | \ 1 \leq j \leq n\} \\
&\leq \max\{\sum_{i=1}^{m} |a_{ij} \times x_i| + |b_i| \quad | \ 1 \leq j \leq n\} && \text{(Triangle Inequality)} \\
&= \max\{\sum_{i=1}^{m} |a_{ij}| \times |x_i| + |b_i| \ | \ 1 \leq j \leq n\} && \text{(Absolute Homogeneity)} \\
&\leq \max\{\sum_{i=1}^{m} |a_{ij}| \times \|\overline{x}\| + \|\overline{b}\| \ | \ 1 \leq j \leq n\} \\
&\leq \max\{\sum_{i=1}^{m} |a_{ij}| \mid 1 \leq j \leq n\} \times \|\overline{x}\| + \|\overline{b}\|,
\end{aligned}$$

so $\|g(\overline{x})\| \in \mathcal{O}(\|\overline{x}\|)$. For any $x \in R$, we have $\mathrm{ReLU}(x) \leq |x|$, so $\mathrm{ReLU}(x) \in \mathcal{O}(x)$. Therefore $\|\mathrm{ReLU}^*(g(\overline{x}))\| \in \mathcal{O}(\|\overline{x}\|)$. Since f is a concatenation of such functions, we also get $\|f(\overline{x})\| \in \mathcal{O}(\|\overline{x}\|)$. □

Lemma 5.1. *Let R be an ordered ring and $f\colon R^* \to R^*$ such that $\sum_{i=1}^{|X|} |f(X)_i| \leq 1$ for all X. Then for all $T \in \mathbf{FT}_R\big(A, B, \mathbf{FF}_R, C, \mathbf{WS}_R(f)\big)$, where A, B and C are arbitrary, we get $\max\{\|T(X)_i\| \mid 1 \leq i \leq |X|\} \in \mathcal{O}(\max\{\|f^{in}(X_i) + f^{pos}(i,n)\| \mid 1 \leq i \leq |X|\})$. Here $\|\overline{x}\| = \max\{|x_i| \mid 1 \leq i \leq |\overline{x}|\}$ denotes the maximum norm.*

Proof. Let $L\colon R^* \to R^*$ be a layer in T using activation function f^{act} and pooling function $f^{\text{pool}} \in \mathbf{WS}_R(f)$. For any X, let $\overline{z}$ be the result of any attention head within $L(X)$. Then

$$\begin{aligned}
\|\overline{z}\| &= \|f^{\text{pool}}(X, \overline{a})\| \\
&= \|\sum_{i=1}^{|X|} f(\overline{a})_i \times X_i\| \\
&\leq \sum_{i=1}^{|X|} \|f(\overline{a})_i \times X_i\| && \text{(Triangle Inequality)} \\
&= \sum_{i=1}^{|X|} |f(\overline{a})_i| \times \|X_i\| && \text{(Absolute Homogeneity)} \\
&\leq \left(\sum_{i=1}^{|X|} |f(\overline{a})_i|\right) \times \max\{\|X_i\| \mid 1 \leq i \leq |X|\} \\
&\leq 1 \times \max\{\|X_i\| \mid 1 \leq i \leq |X|\}. && \text{(Condition on } f\text{)}
\end{aligned}$$

Therefore, when $\overline{z}_1, \ldots, \overline{z}_H$ are the result of the attention heads on vector i, we get

$$\begin{aligned} \|L(X)_i\| &= \|f^{\text{act}}(X_i, \overline{z}_1, \ldots, \overline{z}_H)\| \\ &\in \mathcal{O}(\max\{\|X_i\|, \|\overline{z}_1\|, \ldots, \|\overline{z}_H\|\}) \\ &\subseteq \mathcal{O}(\max\{\|X_i\| \mid 1 \leq i \leq |X|\}). \end{aligned} \qquad \text{(Lemma C.1)}$$

Since T is a concatenation of these layers and the input layer, we also get

$$\|T(X)_i\| \in \mathcal{O}(\max\{\|f^{\text{in}}(X_i) + f^{\text{pos}}(i,n)\| \mid 1 \leq i \leq |X|\}) \quad \text{for all } i \text{ and}$$

$$\max\{\|T(X)_i\| \mid 1 \leq i \leq |X|\} \in \mathcal{O}(\max\{\|f^{\text{in}}(X_i) + f^{\text{pos}}(i,n)\| \mid 1 \leq i \leq |X|\}).$$

□

Definition C.2. We define a transformer $T_{RF}(M)$ by modifying $T_{RC}(M)$. Firstly, we add another layer before the first one. It has two attention heads for each vector coordinate, for computing their maximal and minimal values. For example, we use the attention functions $f^{\text{att}}(\overline{x}, \overline{y}) = x_{one} \cdot \pm y_v$ to compute the maximum and minimum y_v. We can then use ReLU to compute

$$G = c \cdot \max\{\|f^{\text{in}}(X_i) + f^{\text{pos}}(i,n)\| \mid 1 \leq i \leq |X|\}$$

where c is the constant from Lemma 5.1 for $T_{RC}(M)$. This value is stored in an additional coordinate x_G, we replace all occurences of G in Definition B.1 with x_G and redefine how $\otimes$ is computed:

$$a \otimes b = \mathrm{ReLU}(a + b - x_G) - \mathrm{ReLU}(a - b - x_G)$$

The new layer also sets $f_0^{\text{act}}(\overline{x}, \ldots)_{\text{sign}} = c \cdot x_{\text{sign}}$. Finally, the branching conditions $\chi_{\geq}(p_z(x_{var1}, \ldots, x_{varm}), 0)$ in f_4^{act} must now be computed by an attention head using the attention function $f^{\text{att}}(\overline{x}, \overline{y}) = p(x_{var1}, \ldots, x_{varm}) \cdot y_{\text{sign}} \in \mathbf{FCIRC}_R$.

Lemma C.3. *Let R be a $\mathbb{Z}$-ordered ring and M be a linear $\mathbf{BSS}_R$-machine. If X is a sequence of vectors such that the first vector has $x_{\text{sign}} = g$ and the set of x_{sign} in X is $\{g\}$ or $\{0, g\}$ for $g = \max\{\|f^{in}(X_i) + f^{pos}(i,n)\| \mid 1 \leq i \leq |X|\}$, then*

$$\chi_{\geq}(a,b) = \begin{cases} G & \text{if } a \geq b \\ 0 & \text{if } a < b \end{cases}$$

for all $a, b \in R$ with $G = c \cdot g$ within all but the first layer of $T_{RF}(M)(X)$, the first vector of $T_{RF}(M)(X)$ has $x_{\text{sign}} = G$ and the set of x_{sign} in $T_{RF}(M)(X)$ is $\{G\}$.

Proof. After the input layer, the last vector always has $x_{\text{sign}} = 0$, so there is at least one vector with $x_{\text{sign}} = 0$. Then we get $f_0^{\text{act}}(\overline{x}, \ldots)_{\text{sign}} = c \cdot x_{\text{sign}}$, so the set of x_{sign} becomes $\{0, c \cdot g\} = \{0, G\}$. An attention head computing $\chi_{\geq}(a,b)$ for $\overline{x}$ now computes a vector with

$$y_{\text{sign}} = \begin{cases} G & \text{if } x_{\geq} \geq 0 \\ 0 & \text{if } x_{\geq} < 0 \end{cases}$$

As x_{sign} cannot change until the final layer, this works for all layers. In the final layer, we get $f_4^{\text{act}}(x, \overline{y}^{\text{sign}}, \ldots)_{\text{sign}} = G$, so the set of x_{sign} is $\{G\}$. □

Theorem 5.2. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{FCIRC}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *contains all functions computable by linear $\mathbf{BSS}_R$-machines.*

Proof. For all $b \in R$ with $|b| \leq G$, we get $0 \otimes b = \mathrm{ReLU}(0 + b - G) - \mathrm{ReLU}(0 - b - G) = 0 - 0$ and $G \otimes b = \mathrm{ReLU}(G + b - G) - \mathrm{ReLU}(G - b - G) = \mathrm{ReLU}(b) - \mathrm{ReLU}(-b) = b$. Due to Lemma 5.1, we can choose the constant c such that $|b| \leq G$ for all possible values of b within the transformer. Therefore, when using Lemma C.3 instead of Lemma B.2, the proof of Theorem 4.1 also works for T_{RF}. □

Theorem 5.3. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{R\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$ *contains all functions computable by linear $\mathbf{BSS}_R$-machines with degree 2 branching.*

Proof. We perform a slight modification on $T_{RF}(M)$. In layer 3, we set $f_3^{\text{act}}(\overline{x},\dots)_{bi} = x_{\text{sign}} \otimes f_3^{\text{act}}(\overline{x},\dots)_{vari}$ and $f_3^{\text{act}}(\overline{x},\dots)_{bsg} = x_{\text{sign}} \otimes 1$. Then the branching condition $\chi_{\geq}(p_z(x_{var1},\dots,x_{varm}),0)$ in f_4^{act} for a $p_z(x_1,\dots,x_m) = \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}x_ix_j + \sum_{i=1}^{m} b_ix_i + c$ of degree 2 can be computed by an attention head using the following attention function

$$\begin{aligned}
f_z^{\text{att}}(\overline{x},\overline{y}) &= \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}x_{vari}y_{bj} + \sum_{i=1}^{m} b_ix_{vari}y_{bsg} + c\cdot x_{one}y_{bsg} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}x_{vari}(y_{\text{sign}}\otimes y_{varj}) + \sum_{i=1}^{m} b_ix_{vari}(y_{\text{sign}}\otimes 1) + c\cdot x_{one}(y_{\text{sign}}\otimes 1) \\
&= y_{\text{sign}} \otimes \left(\sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}x_{vari}y_{varj} + \sum_{i=1}^{m} b_ix_{vari} + c\cdot x_{one}\right) \\
&= y_{\text{sign}} \otimes p_z(x_{var1},\dots,x_{varm}).
\end{aligned}$$

Therefore, the attention score is 0 for all vectors $\overline{y}$ with $y_{\text{sign}} = 0$ and $p_z(x_{var1},\dots,x_{varm})$ for vectors with $y_{\text{sign}} = G$. So an attention head using f_z^{att} computes $\chi_{\geq}(p_z(x_{var1},\dots,x_{varm}),0)$. □

Definition C.4. We apply the construction from Definition C.2 to $T_{SC}(M)$, but we also need to ensure that all x_{sign} contain only the values 0 and G after layer one. This is done by adding an attention head for $f_{\text{sign}}^{\text{att}}$ to layer one and setting $f_1^{\text{act}}(\overline{x},\overline{y}^{\text{sign}})_{\text{sign}} = \chi_{\geq}(2\cdot x_{\text{sign}},1)\otimes y_{\text{sign}}^{\text{sign}}$. We call the resulant transformer $T_{SF}(M)$.

Lemma C.5. *Let R be a $\mathbb{Z}$-ordered ring and M be a linear $\mathbf{BSS}_R$-machine. If X is a sequence of vectors such that the set of x_{sign} in X is $\{g\}$ or $\{0,g\}$ for $g = \max\{\|f^{in}(X_i) + f^{pos}(i,n)\| \mid 1 \leq i \leq |X|\}$, then*

$$\chi_{\geq}(a,b) = \begin{cases} G & \text{if } a \geq b \\ 0 & \text{if } a < b \end{cases}$$

for all $a,b \in R$ with $G = c\cdot g$ within all but the first layer of $T_{SF}(M)(X)$ and the set of x_{sign} in $T_{SF}(M)(X)$ is $\{G\}$ or $\{0,G\}$.

Proof. Because $c \geq 1$ and $x_p \geq 1$ for all vectors, we always have $G \geq 1$ and therefore either $x_{\text{sign}} = 0$ or $x_{\text{sign}} \geq 1$, if it was set to a previous G.

- When $y_{\text{sign}}^{\text{sign}} = 0$, we get $f_1^{\text{act}}(\overline{x},\overline{y}^{\text{sign}})_{\text{sign}} = 0 = x_{\text{sign}}$ as before.
- When $y_{\text{sign}}^{\text{sign}} > 0$ and $x_{\text{sign}} = 0$, we get $2\cdot x_{\text{sign}} < 1$ and the attention head computing $\chi'_{\geq}(2\cdot x_{\text{sign}},1)$ attends to a vector with $y_{\text{sign}} = 0$.
- When $y_{\text{sign}}^{\text{sign}} > 0$ and $x_{\text{sign}} \geq 1$, we get $2\cdot x_{\text{sign}} \geq 2 > 1$ and the head attends to the vector with the maximum $y_{\text{sign}} = G$.

So the computation of $\chi_{\geq}(2\cdot x_{\text{sign}},1)$ is always correct and the same argument applies to $f_1^{\text{act}}(\overline{x},\overline{y}^p)_{end} = \chi_{\geq}(2\cdot x_p + 1, 2\cdot y_p^p)$, because x_p and y_p^p are positional embeddings.

Therefore, the set of x_{sign} after layer 1 is $\{0,G\}$. The remainder of this proof is a combination of the proofs of Lemmas B.4 and C.3. □

Corollary C.6. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{S\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{FCIRC}_R, \mathbf{WS}_R(\text{left})\big)$ *contains all functions computable by linear $\mathbf{BSS}_R$-machines.*

Proof. By using Lemma C.5, the proof of Theorem 5.2 can also be applied to Theorem 4.2. □

Corollary C.7. *Let R be a $\mathbb{Z}$-ordered ring. Then* $\mathbf{S\text{-}FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\text{left})\big)$ *contains all functions computable by linear $\mathbf{BSS}_R$-machines with degree 2 branching.*

Proof. The construction from Theorem 5.3 also works for $T_{SF}(M)$. □

D On Time Complexity

From our constructions of BSS-machines and transformers we obtain the following time bounds.

Corollary D.1. *Let R be any $\mathbb{Z}$-ordered ring and T be a transformer in $\mathbf{GT}_R\big(\mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{FCIRC}_R[\mathrm{sign}], \mathbf{WS}_R(\mathrm{left})\big)$. Then there is a $\mathbf{BSS}_R$-machine such that if T halts on input X in t recursion steps, then M simulates T in $\mathcal{O}((t+|X|)^3)$ computation steps. This holds for both the **R** and **S** recursion variants.*

Proof. Algorithm 1 simulates a single transformer step on input X in time $\mathcal{O}(|X|^2)$. Because the input length increases by 1 for each step, the simulation of t recursion steps takes time

$$\mathcal{O}\left(\sum_{i=|X|+1}^{|X|+t} i^2\right) = \mathcal{O}\left((|X|+t)^3\right).$$

□

Corollary D.2. *Let R be any $\mathbb{Z}$-ordered ring and M be a $\mathbf{BSS}_R$-machine. Then there is a transformer T in $\mathbf{GT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$, such that if M halts on input X in t computation steps, then T simulates M on input X in $\mathcal{O}(t+|X|)$ recursion steps. This holds for both the **R** and **S** recursion variants.*

Proof. During the simulation of Theorem 4.1, the transformer remains in the input state until the sequence length is at least $\max(2\cdot|X|, \hat{m}+|X|)$ where $\hat{m}$ is a constant depending on M. This takes $\mathcal{O}(|X|)$ recursion steps. The simulation of the output state takes exactly $|Y|$ steps where Y is the output of the BSS-machine and every other state takes only one recursion step. Therefore, the entire simulation takes $\mathcal{O}(|X|+t+|Y|) \subseteq \mathcal{O}(|X|+t)$ recursion steps.

For Theorem 4.2, the steps for the output state must instead be bounded by $s+|Y|$ where s is the number of non-zero tape cells. But since $s \in \mathcal{O}(|X|+t)$, we obtain the same asymptotic bound. □

Since the values within a BSS-machine can be more complex than individual bits, different complexity measures can be defined based on different valuations for them (see Blum et al. [1998]). The simplest kind is using unit-height which defines the size of an input as the number of values, $\mathrm{size}(X) = |X|$.

Definition D.3. Let $\mathrm{size}\colon R^* \to \mathbb{N}$ be a size measure and $f\colon R^* \to R^*$. Then $f \in \mathbf{FP}_R(\mathrm{size})$, if it is computed by a $\mathbf{BSS}_R$-machine that halts after $\mathcal{O}(\mathrm{size}(X)^c)$ steps for some $c \in \mathbb{N}$ and any X.

Definition D.4. Let $\mathrm{size}\colon R^* \to \mathbb{N}$ be a size measure and $\mathfrak{C}$ a class of transformers. Then $f \in \mathbf{PR}(\mathrm{size})\text{-}\mathfrak{C}$, if $f \in \mathbf{R}\text{-}\mathfrak{C}$ via a transformer that halts in $\mathcal{O}(\mathrm{size}(X)^c)$ steps for some $c \in \mathbb{N}$ and any X.

Because the runtime of our constructions depends only on the length of the input and not on the values in it, we get an equivalence for any size measure that is at least as big as unit height.

Theorem 6.1. *Let R be any $\mathbb{Z}$-ordered ring and let* $\mathrm{size}\colon R^* \to \mathbb{N}$ *be any size measure such that* $\mathrm{size}(X) \in \Omega(|X|)$. *Then*

$$\mathbf{FP}_R(\mathrm{size}) = \mathbf{PR}(\mathrm{size})\text{-}\mathbf{FT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big).$$

Proof. Since $\mathrm{size}(X) \in \Omega(|X|)$, we get $|X| \in \mathcal{O}(\mathrm{size}(X)^c)$ for any $c \geq 1$. Then by Corollary D.1, for every function in $\mathbf{PR}(\mathrm{size})\text{-}\mathbf{FT}_R\big(\mathbf{FCIRC}_R, \mathbf{IPE}_R, \mathbf{FCIRC}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big)$, there exists a $\mathbf{BSS}_R$-machine that computes it in time $\mathcal{O}((\mathrm{size}(X)^c+|X|)^3) \subseteq \mathcal{O}((\mathrm{size}(X)^c)^3) = \mathcal{O}(\mathrm{size}(X)^{3c})$.

On the other hand, by Corollary D.2, for every function in $\mathbf{FP}_R(\mathrm{size})$, there exists a transformer that computes it in time $\mathcal{O}(\mathrm{size}(X)^c+|X|) \subseteq \mathcal{O}(\mathrm{size}(X)^c)$. □

Finally, by defining $\mathbf{FP}_R^{\mathrm{L}2}$ as a restriction of $\mathbf{FP}_R$ to linear $\mathbf{BSS}_R$-machines with degree-2 branching, we get the same result for the second equivalence.

Corollary 6.2. *Let R be any $\mathbb{Z}$-ordered ring and let* $\mathrm{size}\colon R^* \to \mathbb{N}$ *be any size measure such that* $\mathrm{size}(X) \in \Omega(|X|)$. *Then*

$$\mathbf{FP}_R^{\mathrm{L}2}(\mathrm{size}) = \mathbf{PR}(\mathrm{size})\text{-}\mathbf{FT}_R\big(\mathbf{FF}_R, \mathbf{IPE}_R, \mathbf{FF}_R, \mathbf{DPA}_R, \mathbf{WS}_R(\mathrm{left})\big).$$

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 paper contains some intermediate results that are not mentioned in the abstract nor the introduction, but all which we deem important are included. However, the statements in the introduction are simplified versions of the results, because stating them in full accuracy requires a lot of definitions.

   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: [N/A]

   Justification: We think that our theoretical results have no limitations other than the assumptions made in the theorems.

   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 necessary assumptions are mentioned either in the theorems directly or in the required definitions. Full proofs for all statements can be found in the appendix.

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: [N/A]

Justification: The paper does not include experiments.

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
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) 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).
  (d) 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: [N/A]

   Justification: The paper does not include experiments.

   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: [N/A]

   Justification: The paper does not include 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: [N/A]

   Justification: The paper does not include experiments.

   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: [N/A]

   Justification: The paper does not include experiments.

   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: We think that none of the requirements in the NeurIPS Code of Ethics apply to our theoretical research.

   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: We think that our theoretical results have no direct impact on society.

    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: We think that our theoretical results pose no such 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: [N/A]

    Justification: The paper does not use existing assets.

    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 any code or dataset.

    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: 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.
    - 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: At no point during the writing of this paper did we use LLMs for any purpose.

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.