KNOWLEDGE TRANSFER BETWEEN SIMPLE DISCRETE WORLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting to a new world and reasoning correctly about which outcomes are possible under rules one has not met before is a basic capacity underlying general intelligence. We study it in small discrete worlds: sets of grid configurations consistent with a shared set of rules, concretely Sudoku-like puzzles built from a common base and a handful of extra rules. We train a single depth-recurrent model on many combinations of rules and test it at four levels of novelty: a new puzzle, a new combination of known rules, a pairing of rules never seen together, and a rule instance not seen at all during training. The model receives the active rules either as a symbolic description or as two solved examples, and a model that gets no rule information is used as a baseline. At recombining rules it was trained on, the description matches or exceeds examples (99% against 95% on new combinations). On rule instances whose components never co-occurred in training, the description falls below the no-information baseline while examples stay above it. The framework allows us to test many such generalization regimes in more systematic way than benchmarks such as ARC-AGI and provides new ground for evaluating depth-recurrent reasoning models, which are typically tested on saturated Sudoku benchmarks.

1 INTRODUCTION

A recurring puzzle in artificial intelligence is the mismatch between domains of competence: systems that exceed human performance on tasks perceived as difficult often fail on tasks that appear far simpler. Following Chollet (2019), this points to a distinction between *narrow intelligence* (high performance on a specific, well-represented task distribution) and *general intelligence* (the capacity to adapt to, and learn the structure of, previously unseen environments). ARC (the Abstraction and Reasoning Corpus, Chollet, 2019) operationalizes this via small sets of input–output examples from which a hidden transformation must be inferred rather than memorized, but its tasks remain shaped by human visual conventions (connected-component segmentation, color-as-category symbolism), so strong performance may partly reflect familiarity with these conventions, which makes such studies hard to control.

We adopt a more minimal formalization, close to the intuitive notion of a *world*: a finite grid governed by a small set of explicit rules, each forbidding certain patterns (Section 2.1 makes this formal). Physical laws can be phrased the same way, and familiar puzzles such as Sudoku and mazes are special cases with a particular rule vocabulary. Understanding a world then means compressing the dependencies between its patterns into a few rules that are simple to store and apply, the core capability of a *world model* (Ha & Schmidhuber, 2018; LeCun, 2022). Because these rules come from the structure of the world itself rather than from a human-designed visual vocabulary, the resulting benchmark is less human-centric than ARC; this matters because general intelligence need not be human-like, and could exceed human intelligence in ways a human-anchored benchmark would fail to capture.

We study how knowledge *transfers* across different worlds, for a relatively small *parameterized* family of worlds: given a model trained on many combinations of rules from a fixed pool, does it correctly apply an unseen *combination* of already-known rules and, more strictly, a rule *instance* it has never encountered? We study this in a family of Sudoku-like grid worlds: a fixed Latin-square base plus a few extra rules drawn from a fixed pool (Section 3). To do so, we train a depth-recurrent

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

model (similar to those recently used on Sudoku (Wang et al., 2025)) on a puzzle together with a context that carries the information about the active rules: either two solved examples from which the rules can be inferred, or the rules stated explicitly as a short symbolic description. A model given no such context serves as the no-information baseline.

We test four levels of novelty, from a new puzzle in a trained world to a rule instance never seen at all. The two ways of communicating the rules turn out equally good at recombining rules the model was trained on. On rule instances whose components were never seen together in training, the symbolic description falls below the no-information baseline while examples stay above it. Section 4 shows how this relates to how much co-occurring rule structure the training set exposes.

The generators from the proposed setup are easily extensible to more complex rule structures and worlds and can therefore replace already saturated fixed-game benchmarks for recurrent reasoning models (Jolicoeur-Martineau, 2025; Davis et al., 2026) such as Sudoku-Extreme (Wang et al., 2025).

Contributions. Code and data are available at an anonymized repository.[1] This paper provides:

1. A formalization of a setup in which several distinct modes of novelty can be studied (Section 2).
2. A parameterized family of Sudoku-like grid worlds whose rules decompose into reusable components (Section 3.1).
3. An empirical characterization of where compositional generalization breaks down and how it depends on how rule identity is communicated (Section 4).
4. Evidence that the limiting factor is exposure to co-occurring rule structure rather than model capacity or the symbolic format (Section 4, Appendix C).

2 Parameterized families of small discrete worlds

This section sets up the formal framework: worlds as sets of configurations avoiding forbidden patterns (Section 2.1), and parameterized families of worlds indexed by combinations of rules from a fixed pool (Section 2.2), instantiated concretely in Section 3.

2.1 Small discrete worlds

Let $\mathcal{A}$ be a finite alphabet and $N \geq 1$ an integer. We call *configuration* any element of $\mathcal{A}^{[\![1,N]\!]^2}$, i.e. any coloring of the $N \times N$ grid with symbols from $\mathcal{A}$. A *pattern* is a function $p : \mathbb{U} \to \mathcal{A}$, where $\mathbb{U} \subset \mathbb{Z}^2$ is a finite set called the *support* of p. A pattern p appears in a configuration $x \in \mathcal{A}^{[\![1,N]\!]^2}$ if some shifted copy of p is found in x: there is $v \in \mathbb{Z}^2$ such that $x_{u+v} = p_u$ for every $u \in \mathbb{U}$.

Definition 2.1 (World)**.** *Let $\mathcal{F}$ be a finite set of patterns on alphabet $\mathcal{A}$. The* world *with parameters $(\mathcal{A}, N, \mathcal{F})$ is $W_N(\mathcal{A}, \mathcal{F}) = \{x \in \mathcal{A}^{[\![1,N]\!]^2} \mid \text{no pattern in } \mathcal{F} \text{ appears in } x\}$. Elements of $W_N(\mathcal{A}, \mathcal{F})$ are called* admissible configurations*, and $\mathcal{F}$ the world's* forbidden patterns*.*

This forbidden-pattern formalization is the finite-grid analogue of the subshift of finite type of symbolic dynamics (Lind & Marcus, 1995). We impose no restriction on pattern support size relative to N: several of our family's rules (Section 3) forbid patterns whose support spans an entire row, column, or box.

2.2 Parameterized families

To study generalization, we consider *families* of worlds sharing the same alphabet and grid size, differing only in which rules are active, and study generalization through different train–test splits over such a family.

Definition 2.2 (Parameterized family)**.** *Let $\mathcal{P} = \{\mathcal{F}_1, \ldots, \mathcal{F}_K\}$ be a finite* pool *of forbidden sets, each $\mathcal{F}_k$ a finite set of patterns defining one* rule*. A* combination *is a subset $S \subseteq \{1, \ldots, K\}$, giving rise to the world $W_N\big(\mathcal{A}, \bigcup_{k \in S} \mathcal{F}_k\big)$. A* parameterized family *associated to $\mathcal{P}$ is any collection $\{\, W_N(\mathcal{A}, \bigcup_{k \in S} \mathcal{F}_k) \mid S \in \mathcal{S} \,\}$, $\mathcal{S} \subseteq 2^{\{1,\ldots,K\}}$, of worlds indexed by some chosen subset $\mathcal{S}$ of all possible combinations of rules from $\mathcal{P}$.*

[1] https://anonymous.4open.science/r/GridWorldHyperNetworks-E993/

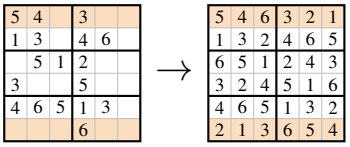


Figure 1: A completion problem (Definition 2.3): given cells D (left) determine a unique admissible configuration x (right) in some world $W_N(\mathcal{A}, \mathcal{F})$. This particular puzzle comes from the concrete family introduced in Section 3.

This shared structure is what makes it meaningful to ask whether a model trained on some worlds generalizes to a new one - a new combination $S \in \mathcal{S}$ of already-known rules, or, more strictly, to a world with a genuinely new rule outside $\mathcal{P}$ altogether.

2.3 Completion problems

Deciding whether a full configuration is admissible is a constraint-satisfaction verification, and every world in our family is a CSP in this sense (Section 5). Beyond that verification, a natural task in a world $W_N(\mathcal{A}, \mathcal{F})$ is to recover a full configuration from a partial one - itself a CSP, of finding a satisfying assignment rather than merely verifying one.

Definition 2.3 (Completion problem)**.** *Let $W_N(\mathcal{A}, \mathcal{F})$ be a world. A* partial configuration *is a pair $(D, \tilde{x})$ with $D \subseteq [\![1, N]\!]^2$ and $\tilde{x} : D \to \mathcal{A}$; cells in D are* given*, cells outside D are* blank*. A partial configuration $(D, \tilde{x})$ is a* completion problem *for $W_N(\mathcal{A}, \mathcal{F})$ if there is exactly one admissible configuration $x \in W_N(\mathcal{A}, \mathcal{F})$ with $x|_D = \tilde{x}$; this unique x is the problem's* solution*, and* solving *it means recovering x from $(D, \tilde{x})$ and the world's rules $\mathcal{F}$. Figure 1 shows one concrete instance.*

Sudoku, and grid puzzles generally, are exactly completion problems for a world whose forbidden patterns encode the puzzle's rules; the fewer cells given, the more the rules alone must determine the rest of the grid, so the size of D is a coarse proxy for difficulty.

3 Experimental setup

We instantiate the framework of Section 2 with a family of Sudoku-like worlds (Section 3.1), three ways of telling the model which rules are active (Section 3.2), four levels of novelty at test time (Section 3.3) and a dial that controls how much compositional structure the training set exposes (Section 3.4). Figure 2 summarizes this setup.

3.1 Sudoku-style parameterized family of worlds

Every world in our family is subject to a fixed set $\mathcal{L}$ of *base* rules, the Latin-square constraint that every row and every column contains each symbol exactly once, together with 2–4 additional rules drawn from a *variable* pool $\mathcal{V}$ of 49 rules; in the terms of Definition 2.2, $\mathcal{P} = \mathcal{L} \cup \mathcal{V}$ and every combination S contains all of $\mathcal{L}$. We use 6×6 grids with symbols $\{1, \ldots, 6\}$, the smallest size divisible by both 2 and 3 as several pool rules tile the grid in 2×3 or 3×2 blocks, which keeps a sweep over thousands of rule combinations tractable without making puzzles trivial (the no-information baseline of Section 4 reaches only 61% in-distribution accuracy on our main split). Nothing in the formalization is specific to this size.

Holding the base fixed serves two purposes. Any generalization gap then reflects the variable pool alone, not a second source of novelty; and because the base constrains every cell through its row and column while the variable rules are mostly local (one box, one diagonal, a pair of rows), many cells can be blanked while a single valid completion remains, and the solver used to sample puzzles can check uniqueness quickly (Section 3.1). The pool rules are chosen for their similarity to the base and to variants familiar from Sudoku (boxes, diagonals, windows), each still introducing some difference, so the pool stays one recognizable family rather than an arbitrary set. Most of the rules

Table 1: Rule types in the 49-instance pool $\mathcal{V}$. Every rule is a set of forbidden patterns on the 6×6 grid; the glyph shades the cells one instance of the type constrains. Full per-instance definitions in Appendix A.

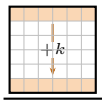
	Type	#	Description of the rules of this type in the family
	Unique-occurrence regions - *a region where each symbol appears at most once*	11	A fixed support (a 2×3 box, its scattered counterpart, a diagonal, a window, the four center or corner cells) contains each symbol at most once.
	Adjacency - *cells in a fixed spatial relation must differ*	2	Two cells a knight's move apart (`AntiKnight`) or diagonally adjacent (`AntiKing`) never share a value.
	Value-restricted regions - *a region confined to a fixed subset of values*	4	Every cell of a diagonal takes values in a fixed set $V \subseteq \{1, \ldots, 6\}$ (odd or even).
	Value-shift relations - *two regions whose values are related by a fixed offset*	32	A pair of rows or columns a, b (one specific pair, or all three pairs of the top and bottom halves) satisfies $b = a + k \bmod 6$ cell by cell, $k \in \{1, \ldots, 5\}$.

Table 2: The three conditioning setups. Only the prefix placed before the 36 puzzle cells differs.

Setup	Prefix	Len.	Symbols used	What the model must do
no-information	none	36	7	Infer the rules from the given cells alone; lower bound on how much rule information the puzzle itself carries.
description-based	4 rules $\times$ 4 tokens	52	36	Read the rule identity as symbols and apply it.
example-based	2 solved grids (72 tokens)	108	7	Infer the rules from the examples, then apply them; mirrors the ARC-AGI format.

in the pool decompose into shared, reusable components (template type, axis, position, parameter), and this lets us hold out a rule while keeping its components in the training set. Table 1 lists the rule types. Appendix F gives the rationale for these design choices.

Puzzles. Not every combination of rules admits a configuration, and a world with none cannot be turned into a puzzle. A combination is retained only if a custom solver (Appendix B) samples enough full configurations from it that stay uniquely completable with at least 14 of the 36 cells blanked, about the fraction blanked in a comparably hard 9×9 Sudoku. To obtain one puzzle we sample an admissible configuration, then blank cells one at a time, rejecting any removal that makes the remainder ambiguous, until a calibrated target blank count is reached. We generate 500 training worlds with 2000 training and 200 held-out puzzles each. The held-out puzzles of 50 of these worlds form the validation set used to select checkpoints, and also serve as the in-distribution test set (level L0 of Section 3.3). The other three levels are evaluated on 30 further worlds each, using their 200 held-out puzzles. Figure 1 shows one such puzzle and its solution.

3.2 How the rules are communicated

A partial configuration alone need not determine its solution when the rules are unknown, so the model must be told, or must infer, which rules are active. We compare three setups (Table 2), differing only in the prefix placed before the 36 puzzle cells.

In the description-based setup, each active rule is represented using four tokens, one for each of the following slots: template type, axis, position, and parameter. If a given rule does not use some slots, or if fewer than four variable rules are present (as worlds have at most four variable rules), the unused slots are filled with a special null token. In this setup, the puzzle prefix has therefore a constant length of 16 tokens. The base rules are never explicitly described. For the world of Figure 1, $\{$`BoxesRect2x3`, `RowLink(1,6,`k=3)$\}$, this description reads

$$\langle \texttt{box}, \varnothing, \texttt{2x3}, \varnothing \rangle \ \langle \texttt{link}, \texttt{row}, \texttt{(1,6)}, \texttt{3} \rangle \ \langle \varnothing, \varnothing, \varnothing, \varnothing \rangle \ \langle \varnothing, \varnothing, \varnothing, \varnothing \rangle$$

(slot names shown here for clarity are schematic; $\varnothing$ is the null token).

The description alphabet adds 29 symbols to the 7 puzzle symbols. In the example-based setup the context is two solved puzzles from the same world. Because the token-mixing layers grow with sequence length, we also report a control with the description padded to the example-based length (Appendix F). Rules fill the four slots in alphabetical order of their identifier (Appendix A).

3.3 Four levels of novelty

A model may generalize to a new puzzle in several qualitatively different ways: the puzzle alone may be new, or the world itself, through a new combination of trained rules, a pairing of rules never seen together, or a rule instance never seen at all (Figure 2c). The parameterized family lets us separate these cases instead of conflating them in one held-out set.

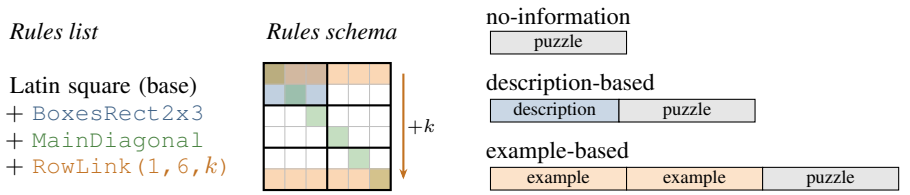
New configuration in a familiar world (L0). The most basic form of novelty is when considering an unseen configuration from a world that was already present during training. This is the standard form of generalization tested by puzzle benchmarks. We report this as our in-distribution reference.

New combination of familiar rules (L1). We next test whether the model can solve puzzles from a world whose complete combination of variable rules was never used during training, while every individual rule in that combination was present in the training set. We refer to this as *combination generalization*.

New co-occurrence of familiar rules (L2). A new rule combination does not necessarily imply that the interactions within it are new. Even if the complete set of active rules has never appeared during training, each of its pairs may have appeared in some other training world. We therefore consider a stronger setting in which whole sets of rule pairs are deliberately prevented from co-occurring during training. We partition the rule pool into two groups and remove from training every world that contains rules from both of them: each rule is still individually represented and pairs within a group still occur, but the model never observes a world in which a cross-group pair is active together. At test time it must solve worlds that contain exactly such previously unseen co-occurrences.

This construction violates the *connected support* condition of Schug et al. (2024), under which any two rules are linked by a chain of training worlds in which consecutive rules co-occur; in this split, which we call *no-mix*, the support falls into two disconnected components. Section 3.4 turns the degree of connectedness into a controlled variable.

New rule instance (L3). Finally, we test generalization to worlds containing a rule instance that was never present during training. The rule pool is parameterized so that many rule instances can be decomposed into components shared with other rules. We can therefore hold out a complete rule while retaining its constituent components elsewhere in the training set. For example, the model sees value-shift rules linking rows 1 and 6 with offsets $k = 1$ and $k = 2$, and offset $k = 3$ on other row pairs, and is tested on the rule linking rows 1 and 6 with $k = 3$. Solving puzzles in such worlds requires applying the structure of a rule type to an instance never encountered as a whole, not merely recombining known rules. Whether this is possible should depend on which of the instance's components were seen together: we call a held-out instance *supported* if every pair of its components (for example, the link type together with the column axis) occurs in some trained rule, and *unsupported* if at least one such pair never occurs in training. Section 3.4 varies the support of a fixed set of held-out instances directly; Appendix C.2 repeats the L3 evaluation on three further held-out sets with three draws each.



(a) example world **(b) how the active rules are communicated**

	L0 (familiar world)	L1 (new combination)	L2 (new pairing)	L3 (new rule instance)
train	$\{A, B, C\}, \{A, D\}$	$\{A, B, C\}, \{A, D\}, \{B, D\}$	$(A, B), (C, D)$	$(A, B), (B, C), (C, D)$
test	$\{A, B, C\}$, new puzzle	$\{A, B, D\}$	(A, C)	(A, H)

(c) forms of novelty

Figure 2: Overview of the setup. (a) Example world: list of rules and schematic representation of these rules. (b) How the rules are communicated: information is appended before the puzzle under diverse forms. (c) Illustration of the forms of novelty. Letters designate rules.

3.4 Controlling exposure to compositional structure

The levels above fix *what* is new at test time. We additionally control *how much* relevant compositional structure the training set exposes, with one dial for L2 and one for L3. The L2 dial starts from the no-mix split and reintroduces training worlds that mix the two rule groups at a graded rate, from none up to the rate at which they occur naturally. The L3 dial holds out a fixed set of column links and varies how many training worlds contain a supporting instance (another column link), from none, where the held-out instances are unsupported, up to the natural rate. In both, the test worlds are identical at every level and the training sets are nested, so any change in accuracy reflects exposure alone, and a no-information model is trained at every level so that each number has its floor beside it. Level values, pair coverages and the full curves are given in Appendix C.1.

3.5 Model and training

All setups share one depth-recurrent model that reads the token sequence and predicts the completed grid. It follows the Hierarchical Reasoning Model recipe (Wang et al., 2025): two coupled latent states updated at different rates, a one-step gradient approximation in place of backpropagation through the full recursion, deep supervision, and a learned halting signal. The block differs: instead of self-attention we use an MLP-Mixer-style block (Tolstikhin et al., 2021), one SwiGLU MLP mixing over sequence positions and one over channels, each with a residual connection and post-RMSNorm. Because the token-mixing weights are tied to absolute positions, no positional encoding is needed. The model has hidden size 256 and 2 blocks, with the same architecture and hyperparameters in every setup; only the input sequence differs, and with it the size of the token-mixing MLP, which gives 1.09M, 1.11M and 1.25M parameters for the no-information, description-based and example-based setups respectively (Appendix F controls for this difference). We train every setup and split with the same AdamW recipe for 20 epochs of 500 batches of 512 puzzles (10,000 batches with 8 optimizer steps each, about five passes over the training puzzles), with a 200-step warmup followed by a constant learning rate of 10^{-4}, keep the checkpoint with the best validation exact-match accuracy, and report exact-match accuracy, the fraction of puzzles whose 36 cells are all correct, as mean $\pm$ standard deviation over 4 training seeds for every run. Where a result is replicated on several draws of a split we report each draw separately, and for the headline comparison we give a paired bootstrap confidence interval over test worlds (Appendix C.2). Appendix E gives the full specification and training details. Figure 2 summarizes this full setup.

4 Results

We compare the three setups of Table 2 at the four novelty levels of Section 3.3, with one model and one training recipe throughout, and place a no-information floor beside every number (Table 3). In

Table 3: Exact-match accuracy in percent, mean ± standard deviation over 4 training seeds, for the three setups at the four novelty levels. L1 is measured on the mixed split and L0 on the held-out puzzles of the validation worlds. L2 is measured on draw 1 of the no-mix split, on worlds containing cross-group pairs. Both L3 columns use the L3 dial split with the same held-out instances and test worlds: unsupported at level 0 (no supporting instance in training), supported at the natural exposure level (228 supporting worlds). Best value per column in bold.

	mixed split		no-mix, draw 1	L3 dial split	
Setup	L0	L1	L2	L3 unsupported	L3 supported
no-information	60.5 ± 0.8	65.4 ± 1.6	63.1 ± 1.8	52.1 ± 2.3	56.7
description-based	$\mathbf{99.2 \pm 0.1}$	$\mathbf{98.5 \pm 0.2}$	67.6 ± 5.5	34.6 ± 6.4	79.0 ± 5.4
example-based	95.2 ± 1.2	95.4 ± 1.1	$\mathbf{82.3 \pm 1.6}$	$\mathbf{65.0 \pm 0.8}$	$\mathbf{90.8 \pm 2.3}$

one sentence: the description-based setup matches or exceeds the example-based setup at recombining trained rules, and falls below the no-information floor on rule instances whose components were never seen together in training.

Recombining trained rules (L0 to L2). On the mixed split the description-based setup reaches 99.2% (L0) and 98.5% (L1), the example-based setup 95.2% and 95.4%, and the no-information floor 60.5% and 65.4%. Under the training budget used, the description is thus not worse than examples at recombining trained rules (here it is even 3 to 4 percentage points better). This ranking depends on the training budget: a shorter schedule reverses it (Appendix C.5). Both setups fit their training puzzles, so neither is capacity-limited. When the training set never crosses the boundary between the two rule groups (L2), both setups fall far below their L1 accuracy: the description to $67.6 \pm 5.5\%$ and examples to $82.3 \pm 1.6\%$. Therefore, rule co-occurrence matters to both setups and the description-based setup depends on it more.

New rule instances (L3). When the held-out instances are supported (every pair of their components occurs in some trained rule), both setups are far above the no-information floor. When they are unsupported, the two setups separate on opposite sides of it (Table 3): the description-based setup drops to 34.6%, 17.5 points *below* the no-information model, while the example-based setup stays 13 points above it. This pattern replicates on six further draws of two other held-out sets: on every draw the description lands 9 to 17 points below the floor and examples 8 to 15 points above it (Appendix C.2). The deficit is not explained by training length, by the amount of training data, or by how the description is injected into the model (Appendix C.4).

Exposure to compositional structure. Both dials add co-occurring structure to the training set while the test worlds stay fixed (Figure 12, Appendix C.1). On the L2 dial, as training worlds that mix the two rule groups are added, both setups climb toward their L1 accuracy while the no-information floor rises only from 62 to 70. The description starts 15 points behind examples and overtakes them once about a fifth of the training worlds mix the groups. On the L3 dial, with identical held-out instances at every level, the description starts 17.5 points below the no-information floor and ends 22 points above it; examples rise too, but by 26 points against 44 for the description. On the splits tested, what limits symbolic composition is therefore whether the components of a new rule were seen together in training, not whether each component was seen by itself. By rule family, the description trails examples at the natural exposure level only on parity rules (Appendix C.6).

Why the description hurts. The pattern is consistent with the description acting as a crutch: a model that receives the correct description on every training puzzle never needs to infer the rules from the grid, so when the description names a pair of components it has never composed, it has nothing to fall back on, whereas the no-information model, forced to infer, does better. Removing the crutch supports this: replacing the description by null tokens on half of the training samples (description dropout) lifts the description-based model above the floor on every setting tried, at a cost of 1 to 2 points on L1 (Table 4; Appendix C.3).

Table 4: Description dropout. Exact match in percent, mean $\pm$ standard deviation over 4 seeds, evaluated with the description present; floor is the no-information model on the same split. The last row shows the cost on in-support rules. Full results, including further splits and evaluation with the description masked, in Appendix C.3.

Setting	description-based	+ dropout	floor	example-based
L3 dial, level 0 (unsupported)	34.6 ± 6.4	58.8 ± 1.6	52.1 ± 2.3	65.0 ± 0.8
no-mix split, L2	67.6 ± 5.5	71.8 ± 2.2	63.1 ± 1.8	82.3 ± 1.6
no-mix split, L3	58.7 ± 3.2	66.6 ± 2.1	61.7 ± 1.6	77.0 ± 2.2
L3 dial, level 0, L1 (in support)	98.3 ± 0.9	96.5 ± 0.2	75.4 ± 0.9	98.0 ± 0.1

5 Related work

Formal definitions of intelligence. Legg and Hutter (Legg & Hutter, 2007) and Hernández-Orallo (Hernández-Orallo, 2017) formalize intelligence as aggregate task performance but specify no canonical, computable task family with explicit difficulty; our parameterized family is a concrete instance, also lining up with Chollet's local/broad spectrum (Chollet, 2019) (extended discussion in Appendix D).

Rule inference. ARC (Chollet, 2019), MiniGrid/BabyAI (Chevalier-Boisvert et al., 2019), CLEVR (Johnson et al., 2017), and Transformer CSP-solvers (Shah et al., 2024; Giannoulis et al., 2025) evaluate rule inference but remain shaped by human-designed vocabulary or a fixed problem family, unlike our combinatorial rule vocabulary. DiG-bench (Battleday et al., 2026) also benchmarks rule discovery, but through long, text-based game transcripts that must be read as large context, a setting not adapted to the implicit, small-recursive-model style of reasoning our setup targets. Every world here is also a structured CSP (Section 2.3) whose regular geometry (Appendix A) is what makes pool rules decompose into the type/axis/position/parameter factors our description-based setup exploits.

Compositional generalization. SCAN (Lake & Baroni, 2018) and COGS (Kim & Linzen, 2020) test a closely related property - generalizing to unseen combinations of familiar components - and find large in-distribution/compositional gaps. Our composition is instead over several constraints that must hold *simultaneously* on a shared grid rather than sequential linguistic structure, and our `L3` tier withholds a rule instance from training *entirely*, stricter than SCAN's primitive split, where the held-out primitive is still demonstrated alone during training. Most closely related, Dziri et al. (2023) find compositional generalization on multiplication/logic-grid puzzles collapses with *computation-graph* depth (*linearized subgraph matching* rather than a general algorithm); their complexity axis is instance *size*, ours *which rules govern the instance*. Schug et al. (2024) show that modular solutions that generalize compositionally require a training distribution with connected support over modules; our no-mix split violates that condition by construction, and our exposure dials measure how much co-occurrence beyond it is needed, separately for symbolic descriptions and for demonstrations.

6 Limitations and further research

- **Exploring other applications of the framework.** We instantiate the abstract framework of Section 2 with one concrete family of Sudoku-style grid worlds, but nothing in the formalism is specific to this choice - other parameterized families, over other rule vocabularies and grid structures, could be built the same way. The construction also extends naturally to space-time trajectories of a discrete dynamical system by adding a time axis, so that forbidden patterns span both space and time; exploring such dynamical instances is left to future work.
- **Only one family of implicit reasoning models tested.** All experiments use one HRM-style architecture (Section 3.5); we have not tested other variants of implicit, recurrent reasoning models along the same rule-adaptation dimension studied here, nor compared against large language models, models using explicit reasoning, or agentic frameworks, which could in principle approach these tasks very differently. Whether the conclusions of Section 4 transfer to these other classes of models is untested.

- **Variations of the design choices.** Several of our design choices are held fixed across every world and setup even though the right value could plausibly vary with the specific rule combination; we did not systematically explore this. For example, the example-based setup always shows $M = 2$ worked examples, a default carried over from an earlier version of the design rather than derived from a per-world or per-rule-type identifiability argument. Earlier experimentation varying M suggested that how many examples are actually needed to disambiguate a world's rule combination depends on the world itself, so a single fixed M may be more than necessary for some worlds and too few for others; characterizing this per-world requirement is left to future work.

Appendix C.7 lists further limitations of the evidence.

7 Conclusion

We proposed a framework for studying adaptation to new worlds in a controlled way, where a world is defined as a finite grid with a set of forbidden patterns. We create a family of worlds by choosing which rules from a fixed pool are active. At test time, we can control the type of novelty: giving the model a new puzzle in a familiar world, a new combination of trained rules, a pairing of rules that were never seen together, or a rule that was never seen in training. Because the rules decompose into reusable components, the framework also lets one control how much co-occurring structure the training set exposes while holding the test worlds fixed, and lets one compare ways of communicating the active rules (a symbolic description or solved examples) under identical conditions. The formalism is not tied to the Sudoku-like family instantiated here. Other rule vocabularies, grid structures, and, with a time axis, discrete dynamical systems fit the same definitions.

As a first use of the framework, we trained one depth-recurrent model across many combinations of rules and measured its accuracy at the four novelty levels. The experiments showed that recombining trained rules is easy whichever way the rules are communicated. Applying a rule whose components were never seen together is hard for both, and here the symbolic description does worse than no description at all. Adding co-occurring structure to the training set, or replacing the description by null tokens on half of the training samples (description dropout), repairs this. To adapt to a new world, a model needs to see not just each component of a new rule on its own, but also see those components together. Our framework makes it possible to measure and control how much of this combined information the model sees in training.

AI Usage Statement

AI tools were used throughout the research and writing process as collaborative assistants. The main research ideas and conceptual motivation originated from the authors. AI was subsequently used to help operationalize these ideas, develop and execute parts of the experimental workflow, and draft portions of the manuscript.

In particular, Claude Code was used iteratively to assist with implementation and experimentation. The authors continuously reviewed, discussed, and evaluated the code and proposed approaches, providing feedback and refining the direction of the work throughout this process. Thus, the use of AI involved an iterative feedback loop between the authors and the AI system instead of autonomous generation of the research.

AI tools were also used to assist in drafting sections of the paper. Drafts were repeatedly reviewed and revised by the authors, who corrected inaccurate or misleading statements, requested modifications, incorporated additional ideas, and determined the final content and interpretation. Code, experimental procedures, results, mathematical statements, and citations suggested or generated with the assistance of AI were checked by the authors before being incorporated into the work.

The authors are responsible for the final methodology, claims, analysis, interpretation, and presentation of the work.

REFERENCES

Ruairidh M. Battleday, Kai Sandbrink, Jimi Cullen-Drohan, Zihan Yan, Timothy Muller, Clare Maguire, Ales Kubicek, Fraser Greenlee-Scott, Sukrit Sumant, Tri Dao, Jürgen Schmidhuber, Michal Valko, Joshua Tenenbaum, Thomas L. Griffiths, Zeb Kurth-Nelson, and James C. R. Whittington. DiG-bench: Discovery in games. *arXiv preprint arXiv:2608.12593*, 2026.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Sahota, Andrew Nguyen, Minh Quan Dai, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *ICLR*, 2019.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Liam Davis, Leopold Haller, Alberto Alfarano, and Mark Santolucito. Lattice deduction transformers. *arXiv preprint arXiv:2605.08605*, 2026.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Panagiotis Giannoulis, Yorgos Pantis, and Christos Tzamos. Teaching transformers to solve combinatorial problems through efficient trial & error. *arXiv preprint arXiv:2509.22023*, 2025.

Judah Goldfeder, Philippe Wyder, Yann LeCun, and Ravid Shwartz Ziv. Ai must embrace specialization via superhuman adaptable intelligence, 2026. URL https://arxiv.org/abs/2602.23643.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

José Hernández-Orallo. *The Measure of All Minds: Evaluating Natural and Artificial Intelligence*. Cambridge University Press, 2017.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, 2018.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022.

Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444, 2007.

Douglas Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

Simon Schug, Seijin Kobayashi, Yassir Akram, Maciej Wołczyk, Alexandra Proca, Johannes von Oswald, Razvan Pascanu, João Sacramento, and Angelika Steger. Discovering modular solutions that generalize compositionally. In *International Conference on Learning Representations (ICLR)*, 2024.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. *arXiv preprint arXiv:2409.10502*, 2024.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

APPENDIX

This appendix is organized into self-contained parts, each expanding on one aspect of the main text: Appendix A lists every rule in the pool; Appendix B details how puzzles are generated and solved; Appendix C gives the exposure dials, the further held-out sets, the description-dropout results, all controls, the provenance of an earlier result, the breakdown by rule family, and further limitations of the evidence; Appendix D expands on two discussions of the main text; Appendix E gives the full architecture and training specification; and Appendix F gives the rationale behind task and data design choices. Each part can be read on its own.

A FULL RULE-POOL DEFINITIONS

This appendix lists every rule in $\mathcal{V}$, the fixed 49-instance pool from which each world's 2–4 variable rules are drawn (Section 3.1), grouped by the three families already introduced in Table 1.

Recall that the grid is 6×6, with symbols $\{1, \ldots, 6\}$. Every puzzle is always subject to the base Latin-square constraint (each row and each column contains all six symbols exactly once); on top of this base, each specific world adds 2–4 extra rules drawn from $\mathcal{V}$.

- **Box variants:** `BoxesRect2x3`, `BoxesRect3x2`: the grid is cut into 2×3 or 3×2 boxes, and each box must contain every symbol exactly once (a rectangular generalization of the classic 3×3 Sudoku box rule). Their scattered counterparts, `ScatteredRect2x3`/`ScatteredRect3x2`, group cells across boxes instead of within them: for each of the 6 positions a cell can occupy inside a box (e.g. its top-left corner), the 6 cells at that position - one from each of the grid's 6 boxes - must together contain every symbol exactly once. Figure 3 illustrates both.

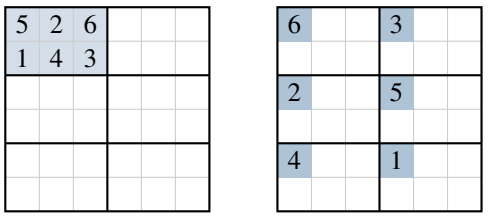


Figure 3: Left: the `BoxesRect2x3` rule requires the six cells physically inside one 2×3 box (shaded, one possible filling shown) to together contain every symbol exactly once. Right: its scattered counterpart `ScatteredRect2x3` instead groups the six cells that share the same position *across* the grid's six boxes (here, each box's top-left corner, shaded) into one such group.

- **Diagonal variants:** `MainDiagonal` (the $\searrow$ diagonal) and `AntiDiagonal` (the $\swarrow$ diagonal) each independently require a unique occurrence of each symbol across their six cells, kept as separate rule instances rather than bundled into one (one of several design choices that makes this pool more *factorizable* into reusable, independently-composable pieces). Figure 4 illustrates both.

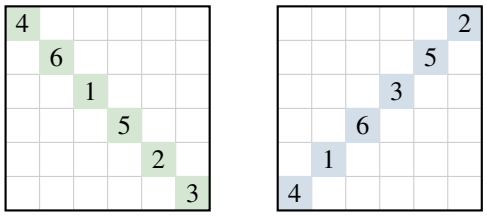


Figure 4: Left: `MainDiagonal` requires the six cells on the $\searrow$ diagonal (shaded, one possible filling shown) to together contain every symbol exactly once. Right: `AntiDiagonal` imposes the same requirement on the six cells of the $\swarrow$ diagonal instead.

- **Adjacency constraints:** `AntiKnight` forbids any two cells a chess knight's-move apart from sharing a value; `AntiKing` forbids any two diagonally-adjacent cells from sharing a value. Figure 5 illustrates both.

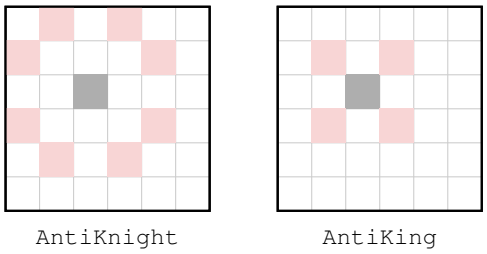


Figure 5: Left: `AntiKnight` forbids the highlighted cell (dark grey) from sharing a value with any cell a chess knight's-move away (red). Right: `AntiKing` forbids the same cell from sharing a value with any diagonally-adjacent cell (red).

- **Parity-on-diagonal rules:** `ParityDiagOdd`/`ParityDiagEven` restrict every cell on the main diagonal to only odd values $\{1, 3, 5\}$ or only even values $\{2, 4, 6\}$, respectively; `ParityADiagOdd`/`ParityADiagEven` apply the same restriction to the anti-diagonal instead. Figure 6 illustrates two of the four combinations.

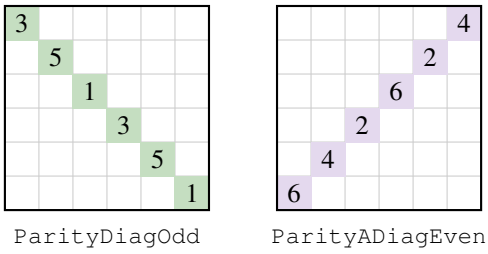


Figure 6: Left: `ParityDiagOdd` restricts every cell on the main diagonal (shaded) to the odd values $\{1, 3, 5\}$; unlike the earlier diagonal rule, repeats are allowed here, since membership in this fixed set is all that is required. Right: `ParityADiagEven` applies the analogous restriction to the anti-diagonal, with the even set $\{2, 4, 6\}$. The remaining two combinations, `ParityDiagEven` and `ParityADiagOdd`, swap which parity set applies to which diagonal.

- **Windows:** `Windows6` places a pair of extra 2×3 "window" regions at a fixed pair of corner offsets in the grid, each independently a unique-occurrence group, offset from the `BoxesRect2x3` box tiling so that it straddles box boundaries, in the spirit of a real

Windoku puzzle. `WindowsWrapH` and `WindowsWrapV` extend this idea by wrapping a window pair around the grid's border: `WindowsWrapH` takes a 3-column band that wraps from the last column back to the first (columns $6, 1, 2$), instantiated at two different row positions; `WindowsWrapV` takes a 2-row band that wraps from the last row back to the first (rows $6, 1$), instantiated at two different column positions. Figure 7 shows all three, with the box tiling overlaid for reference.

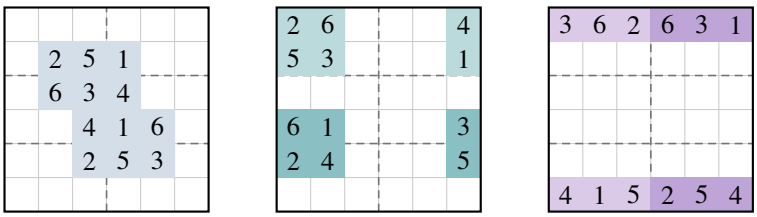


Figure 7: The three window placements (shaded, one possible filling shown), with the 2×3 box tiling from Figure 3 overlaid as a dashed reference grid. `Windows6`'s pair straddles the box boundaries, in the spirit of a real Windoku puzzle. `WindowsWrapH` and `WindowsWrapV` instead each wrap a window pair around the grid's border (matching shade intensity marks the two pieces of the same window, split across the wrap); neither ever coincides with a `BoxesRect2x3` box.

- **Shift rules with explicit offset k:** `RowShiftK{k}` cuts the grid into two fixed areas, the top three rows and the bottom three rows, and requires that, position by position, the value in the bottom area equals the value in the top area shifted by k in the alphabet (mod 6) - the two areas themselves never change with k, only how far the alphabet value is shifted between them. `ColShiftK{k}` is the column-wise analogue, cutting the grid into a left and a right area instead. Figure 8 illustrates this for general k. In $\mathcal{V}$, only $k = 3$ is used for these particular rules; $k = 2$ and $k = 4$ are provably unsatisfiable for this rule (informally: each base column's six linked cells must split evenly into a set S and its shift $S + k$, which requires the shift map's orbits on $\{1, \ldots, 6\}$ - of size $6/\gcd(k, 6)$ - to be even; that size is the odd number 3 exactly for $k \in \{2, 4\}$, hence the conflict, while it is 6 or 2 for $k \in \{1, 3, 5\}$), and $k = 5$, while satisfiable in isolation, admits no configuration in combination with any other pool rule, so neither was added.

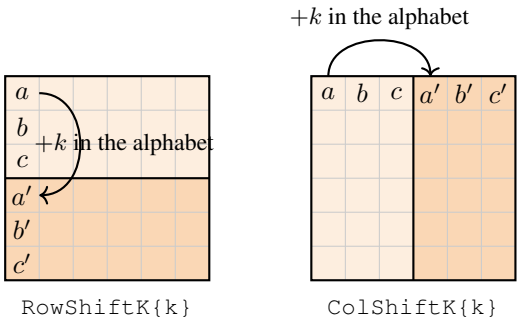


Figure 8: Both rules cut the grid into two fixed areas and then shift the *alphabet value*, position by position, by k between them - the areas themselves never move or depend on k (x' denotes x shifted by k in the alphabet, mod 6). Left: `RowShiftK{k}` cuts the grid into a top and a bottom area (one column shown: top values a, b, c; the corresponding bottom values must be a', b', c'); every other column is cut and shifted the same way. Right: `ColShiftK{k}` is the column-wise analogue, cutting a left and a right area instead.

- **Link rules** (30 of the 49 instances): `RowLink{top,bottom,k}` / `ColLink{left,right,k}` each link one specific pair of rows (or columns)

with a value shift of k, cell by cell. $\mathcal{V}$ includes every such pair at $k \in \{1, 2, 3, 4, 5\}$; unlike the shift-rule family above, constraining only *one* pair at a time avoids the orbit-parity conflict, so all four non-trivial k values admit configurations here even though $k = 2, 4$ are unsatisfiable for `RowShiftK`/`ColShiftK`. This duplication is deliberate: each component (axis, pair of indices, offset k) appears in several contexts, so that a rule instance never seen as a whole can still be represented from its components. Figure 9 illustrates one row pair and one column pair.

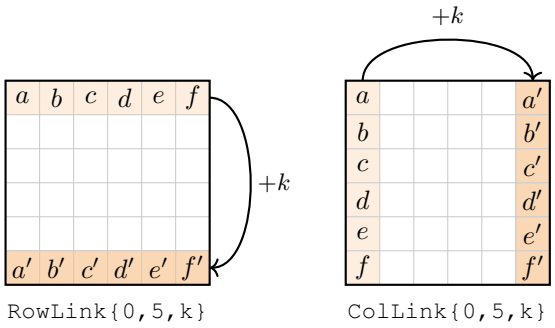


Figure 9: Left: `RowLink{0,5,k}` links only row 1 and row 6 (shaded) - every column's value in row 6 must equal its value in row 1 shifted by k in the alphabet (x' denotes $x+k \bmod 6$); every other row is unconstrained by this rule. $\mathcal{V}$ includes two more such row pairs, $(2, 5)$ and $(3, 4)$ (1-indexed), and the column analogues, each at any $k \in \{1, \ldots, 5\}$ as a separate instance. Right: `ColLink{0,5,k}` is the column-wise version, linking only column 1 and column 6.

- **Fixed small regions:** `Center4` requires the four center cells of the grid to all differ from each other; `Corners4` requires the same among the four corner cells. Neither requires every symbol to appear - with only 4 cells and 6 symbols, this is a plain all-different constraint, not a unique-occurrence region like the box/window/diagonal families above. Figure 10 illustrates both.

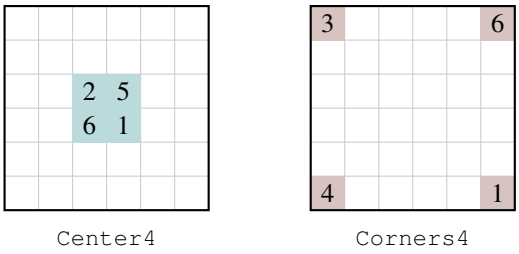


Figure 10: Left: `Center4` requires the four center cells (shaded) to all differ from each other - only 4 of the 6 symbols are involved, so unlike the region rules above, no symbol is required to appear. Right: `Corners4` imposes the same all-different requirement on the four corner cells instead.

The families above are not an exhaustive catalogue of *natural* Sudoku-style constraints. For example, every rule here that relates two rows or columns (the shift and link families) relates their *alphabet values* position by position, never their *positions* themselves; a rule requiring row 6 to equal row 1 *circularly shifted by some number of columns* - a positional, grid-wise rotation, rather than a value-wise one - would be an equally natural variant not present in $\mathcal{V}$.

B Puzzle generation and solver details

This appendix provides all details of puzzle-generation described succinctly in Section 3.1. Figure 11 shows two admissible configurations of one concrete world, of the kind this pipeline samples from before digging a puzzle out of one of them.

2	6	4	3	5	1
3	1	5	4	6	2
4	5	6	2	1	3
1	2	3	5	4	6
6	4	2	1	3	5
5	3	1	6	2	4

configuration 1

3	2	5	1	4	6
4	6	1	5	3	2
2	1	6	3	5	4
5	3	4	2	6	1
1	4	3	6	2	5
6	5	2	4	1	3

configuration 2

Figure 11: Two admissible configurations of the world {`BoxesRect2x3`, `RowLink(1,6,`k=3)} on top of the Latin-square base (thick lines mark boxes; shaded rows are linked, row 6 equals row 1 shifted by 3). These are the two solved grids the example-based setup (Section 3.2) shows the model before a puzzle from this same world (Figure 1 shows one such puzzle).

Non-emptiness check. Not every combination of rules admits a configuration. For each candidate combination the generator samples full admissible configurations within a fixed time budget and digs a puzzle out of each one (see below). The combination becomes a world only if enough configurations were found and, in the median, at least 14 of the 36 cells could be blanked while keeping the solution unique, about the fraction blanked in a comparably hard 9×9 Sudoku. This is an empirical filter, not a proof of non-emptiness.

Generation pipeline. Each puzzle is generated as follows:

1. **Solving:** sample a valid configuration satisfying the world's constraints (base + extra rules) - a complete assignment with zero blanks (using the constraint-propagation and backtracking solver described below).
2. **Digging:** starting from that full configuration, remove cells one at a time (setting them to blank), checking after each removal (using the solver again) that the remaining partial grid still has only one valid completion. If removing a cell would make the puzzle ambiguous (more than one grid fits the remaining clues), that removal is rejected and a different cell is tried.
3. **Stopping at a target difficulty:** digging continues until the puzzle reaches a target number of blanks (a range calibrated per world by a separate search for a blank-count range that the world supports). More blanks generally means fewer given clues and a harder puzzle to solve, since more constraint-propagation is needed to pin down the unique answer. Reaching the target is not guaranteed: digging makes a single pass over the cells in one random order, and if that order is exhausted before the target is reached - because every remaining given cell is now needed for uniqueness - the entire attempt is discarded and generation restarts from a freshly sampled full configuration, rather than retrying the same one with a different cell order.

Solver. Solving and uniqueness checking use a standard constraint-propagation solver with backtracking: cell domains are tracked as bitmasks and shrunk by naked and hidden singles and by the pool's binary relations (not-equal, not-consecutive, equal-shifted-by-k); a minimum-remaining-values search resolves the rest, and uniqueness is established by counting solutions up to two. This is fast enough for 6×6 grids; larger grids would call for a SAT solver.

Dataset size per world. We generate 2,000 training and 200 held-out puzzles per world (Section 3.1).

C Additional results

All numbers are exact-match accuracy in percent under the recipe of Section 3.5 unless a table says "screening recipe" (12 epochs with cosine decay, used only for the early mechanism screen). "Floor" is the no-information model trained on the same split.

C.1 EXPOSURE DIALS

Figure 12 shows both dials of Section 3.4; Tables 5 and 6 give the numbers. In the L2 dial, ρ is the fraction of training worlds containing at least one cross-group pair, and the pair coverage $c(\rho)$ is the fraction of cross-group pairs occurring in the test worlds that occur in at least one training world; the levels are 0, 5, 25, 100 and 373 mixing worlds out of 500, the last being the natural rate. In the L3 dial, five column links are held out at every level and the levels are 0, 5, 25, 100 and 228 training worlds containing one of the ten remaining column links, the last again being the natural rate. In both dials the test worlds are identical at every level and the training sets are nested. The L1 columns of Table 6 show that the runs whose L3 accuracy moves by 45 points are tied at L1 at every level.

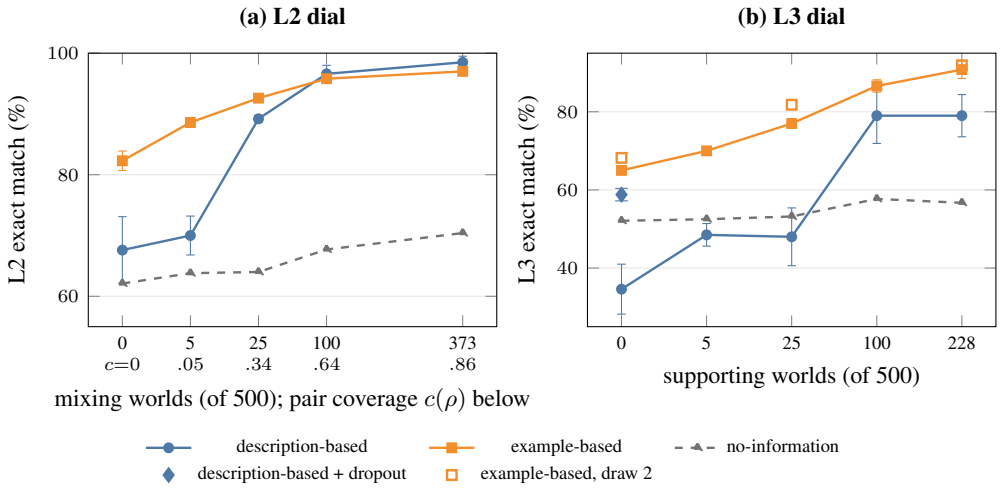


Figure 12: Exposure dials (Section 3.4). (a) L2 accuracy on draw 1 of the no-mix split as training worlds mixing the two rule groups are added; the test worlds are the same at every level and the training sets are nested. (b) L3 accuracy on the held-out column links as training worlds containing a supporting column link are added; held-out instances and test worlds are the same at every level. Points are means over 4 seeds, bars are one standard deviation. The diamond is the description-based model trained with description dropout; hollow squares are the example-based setup on a second draw of the training worlds.

Table 5: L2 exposure dial, draw 1 of the no-mix split. Exact match in percent, mean $\pm$ standard deviation over 4 seeds. Test sets are identical at every level; training sets are nested. L0 accuracy is 97.8 to 99.5% for the description-based and example-based setups at every level.

mixing	ρ	$c(\rho)$	L2			L3		
worlds			description-based	example-based	floor	description-based	example-based	floor
0	0	0.00	67.6 ± 5.5	82.3 ± 1.6	62.1	58.7 ± 3.2	77.0 ± 2.2	59.7
5	0.01	0.05	70.0 ± 3.2	88.6 ± 0.8	63.8	63.6 ± 4.8	79.3 ± 0.9	61.9
25	0.05	0.34	89.2 ± 0.6	92.6 ± 0.5	64.0	70.5 ± 2.5	78.8 ± 1.4	62.4
100	0.20	0.64	96.6 ± 1.4	95.8 ± 0.3	67.7	73.4 ± 0.8	81.8 ± 0.3	61.2
373	0.746	0.86	98.5 ± 1.0	97.0 ± 0.3	70.4	77.8 ± 3.3	82.0 ± 1.4	60.9

C.2 FURTHER HELD-OUT SETS FOR L3

Besides the L3 dial, we evaluate L3 on three further held-out sets, named by what their held-out instances have in common. The *mixed* split holds out rule instances spread across the pool, most of them supported. The *axis* split holds out every column link, so that no trained rule combines the link type with the column axis, while row links remain in training. The *row-pair* split holds out the row links at one pair of row positions, so that no trained rule combines the link type with that position pair.

Table 6: L3 exposure dial, draw 1. Five column links are held out at every level; the level is the number of training worlds containing one of the ten remaining column links (supporting instances). Exact match in percent, mean ± standard deviation over 4 seeds. Test sets are identical at every level. Second training draw at levels 0, 25 and 228 (2 seeds): example-based 68.2 ± 0.3, 81.8 ± 2.1, 91.9 ± 1.4; description-based 39.8 ± 0.2, 69.2 ± 1.8, 84.4 ± 3.3.

support	supporting	L3			L1		
worlds	instances (of 10)	description-based	example-based	floor	description-based	example-based	floor
0	0	34.6 ± 6.4	65.0 ± 0.8	52.1	98.3 ± 0.9	98.0 ± 0.1	75.4
5	5	48.5 ± 2.9	70.0 ± 0.6	52.5	98.5 ± 0.6	98.1 ± 0.3	75.5
25	10	48.0 ± 7.4	77.0 ± 1.1	53.2	98.4 ± 0.1	97.9 ± 0.2	74.1
100	10	79.0 ± 7.1	86.6 ± 1.6	57.7	98.2 ± 0.4	97.8 ± 0.3	74.0
228	10	79.0 ± 5.4	90.8 ± 2.3	56.7	97.8 ± 0.9	97.6 ± 0.2	68.8

For the axis, row-pair and no-mix splits we make three independent draws of training and test worlds; the mixed split has one draw. The no-mix split separates the value-shift rules (the link and shift families of Appendix A) from all other rules. Table 7 gives every draw. On every axis and row-pair draw the description-based setup scores 9 to 17 points below the floor and the example-based setup 8 to 15 points above it, while both are tied with each other at L1 in the same runs.

A paired bootstrap over test worlds on draw 1 (per-world accuracy averaged over all seeds of a setup, 10,000 resamples) gives, for the description-based setup minus the floor, -9.0 with a 95% interval of $[-13.0, -4.9]$ on the axis split and -12.8 $[-18.7, -6.8]$ on the row-pair split; on no-mix L3 the same difference is -2.9 $[-10.6, +4.4]$, so its interval includes zero.

Table 7: Further held-out sets for L3, three draws each. Each cell gives L3 / L1 exact match in percent (for the no-mix split, L3 / L2). Mean ± standard deviation over 4 seeds. The mixed split has one draw: description-based 73.4 ± 2.1, example-based 81.2 ± 0.9, floor 55.9 ± 0.3 on L3.

Split	Draw	description-based	example-based	floor
axis	1	36.4 ± 3.7 / 94.0	59.4 ± 1.8 / 97.0	45.4 / 66.3
	2	37.3 ± 3.2 / 98.6	60.3 ± 1.4 / 97.8	50.4 / 69.5
	3	35.8 ± 2.1 / 98.0	64.0 ± 0.2 / 97.8	50.8 / 76.8
row-pair	1	44.9 ± 3.5 / 97.3	72.2 ± 0.5 / 97.0	57.7 / 67.4
	2	41.4 ± 4.8 / 97.8	69.6 ± 1.3 / 96.4	55.9 / 68.6
	3	44.9 ± 0.4 / 98.8	69.4 ± 1.6 / 97.5	61.6 / 66.8
no-mix (L3 / L2)	1	58.7 ± 3.2 / 67.6 ± 5.5	77.0 ± 2.2 / 82.3 ± 1.6	61.7 / 63.1
	2	59.7 ± 1.7 / 67.6 ± 2.1	74.5 ± 2.6 / 78.1 ± 4.5	57.0 / 56.5
	3	57.6 ± 0.8 / 66.4 ± 2.5	74.2 ± 2.4 / 79.5 ± 3.7	58.0 / 59.4

C.3 Description dropout

To test whether the description-based setup's deficit on unsupported instances comes from never having to infer rules from the grid, we train a variant in which the description prefix is replaced by null tokens on a random half of the training samples ($p = 0.5$). Such a model is evaluated both with the description present and with it masked; masked, it should behave like a no-information model of the same sequence length. Tables 8 and 9 report both, on single splits. Three facts follow. First, an ordinary description-based model uses its description: masked, it falls to 28 to 33% on L3 at every level of the L3 dial and to about 40% on L1 (Table 9; masked L0 is 27 to 36% on every split and level). Second, dropout lifts the description-based model above the floor on every setting tried, at a cost of 1 to 2 points on L1. Third, on the L3 dial at level 0, the dropout model evaluated with the description masked reproduces the 52-token no-information model to within 2 points on L3, L1 and L0, so the dropout model is a no-information model plus a description that only adds.

Table 8: Description dropout, evaluated with the description present. Exact match in percent, mean ± standard deviation over 4 seeds. The floor column gives the no-information model at 36 tokens and, after the slash, at 52 tokens (padded to the description-based sequence length).

Setting	description-based	+ dropout	floor (36 / 52 tokens)	example-based
L3 dial, level 0 (unsupported)	34.6 ± 6.4	58.8 ± 1.6	52.1 ± 2.3 / 53.2 ± 0.4	65.0 ± 0.8
axis draw 1 (unsupported)	36.4 ± 3.7	53.9 ± 2.2	45.4 ± 0.1 / 46.0 ± 2.0	59.4 ± 1.8
no-mix draw 1, L2	67.6 ± 5.5	71.8 ± 2.2	63.1 ± 1.8 / 64.0 ± 1.3	82.3 ± 1.6
no-mix draw 1, L3	58.7 ± 3.2	66.6 ± 2.1	61.7 ± 1.6 / 62.9 ± 0.2	77.0 ± 2.2
L3 dial, level 0, L1	98.3 ± 0.9	96.5 ± 0.2	75.4 ± 0.9 / 77.7 ± 0.5	98.0 ± 0.1

Table 9: Evaluation with the description replaced by null tokens at test time, exact match in percent. Mean ± standard deviation over 4 seeds. On the L3 dial at level 0, the dropout model matches the 52-token no-information model to within 2 points; the ordinary description-based model collapses.

Setting	Level	description-based, masked	+ dropout, masked	floor at 52 tokens
L3 dial, level 0	L3	28.0 ± 2.9	52.4 ± 2.0	53.2
	L1	37.4 ± 4.8	78.4 ± 0.6	77.7
	L0	33.3 ± 3.8	74.7 ± 0.5	72.8

C.4 Controls

Table 10 gives the sequence-length and mechanism controls and Table 11 the mechanism screen. Padding the description-based prefix to the example-based length (108 tokens, the same model size) makes the axis deficit worse and removes most of the row-pair deficit, so sequence length is ruled out as the cause on the first split but not on the second. Further controls, draw 1 of the axis split unless noted:

- **Halting.** Running all 32 recurrent rounds instead of halting, over 50 checkpoints and their L1 and L3 cells, changes the cell means by at most 1.4 points in any setup and never raises them by more than 0.5.
- **Training length.** Per-epoch L3 accuracy of the description-based setup stays between 28 and 40% from the first epoch on, with no upward trend; training accuracy passes 95% at epoch 4 and 99% at epoch 9, and L1 climbs to 94.6%. Training for 40 epochs gives $34.4 \pm 2.8\%$ on L3, 94.5% on L1 and 99.7% on L0.
- **Training data.** With 1,000 training worlds instead of 500 and the same 10,000 batches: description-based $43.1 \pm 3.1\%$ (L1 98.0%), floor $47.2 \pm 0.4\%$ (L1 67.7%).
- **Injection mechanism.** Five ways of injecting the description were screened on the mixed split (Table 11); the three that train are within 2 points on L1, and the best of them, composed rule vectors, gives $36.0 \pm 5.8\%$ on the axis split (Table 10). In the design used in the main text the description is a prefix of the input: four tokens per rule, so 16 description tokens placed before the 36 puzzle cells and processed like any other input. The alternatives leave that prefix empty (null tokens) and hand the description to the model separately; the two that train are described below, and two further ones (a spatial field of learned grid patterns and FiLM-style per-block modulation) did not converge under the screening recipe (Table 11). All five run at 108 positions under the screening recipe, and every added component starts as the identity.
  - *Composed rule vectors.* A small network first merges the four tokens of each rule into one vector, so the description becomes four vectors, one per rule, added to the input at the first four positions. This tests whether the difficulty is keeping track of which tokens belong to the same rule.
  - *Cross-attention.* After each block, every one of the 36 cells attends to the description tokens and reads from them, choosing by content which rule information to use, instead of relying on fixed position-to-position mixing weights.

Table 10: Length and mechanism controls, draw 1 of each split, L3 exact match in percent, 4 seeds. Parameter counts: 1.09M at 36 tokens, 1.11M at 52, 1.25M at 108. Padding the description-based prefix to the example-based length moves the three hard splits by -8.5, $+10.4$ and -0.2 points, with no consistent sign. [a]Screening recipe, 2 seeds; the unpadded reference at that recipe is 67.9. [b]L2 on the same split: 58.1 ± 8.2. [c]Screening recipe, 2 seeds.

Control	axis	row-pair	no-mix	mixed
description-based at 52 tokens (reference)	36.4	44.9	58.7	73.4
description-based padded to 108 tokens	27.9 ± 1.6	55.3 ± 5.0	58.5 ± 1.4	67.5 ± 3.8[a]
no-information at 36 tokens	45.4	57.7	61.7	55.9
no-information at 52 tokens	46.0 ± 2.0	58.8 ± 0.0	62.9 ± 0.2	not run
composed rule vectors	36.0 ± 5.8	57.9 ± 10.2	53.7 ± 5.1[b]	70.8[c]

Table 11: Injection mechanisms for the rule description, mixed split, screening recipe (12 epochs, cosine decay), 2 seeds. Exact match in percent at L0 / L1 / L3. The three working mechanisms are within 2 points of each other on L1.

Mechanism	L0	L1	L3
composed rule vectors	99.1	98.3	70.8
cross-attention from cells to description tokens	98.5	97.2	65.6
description tokens in the sequence (padded)	99.0	96.2	67.5
spatial field (did not converge)	92.4	91.5	60.5
generated per-layer modulation (FiLM; failed)	69.9	73.7	57.7

C.5 Provenance of the earlier L1 result

An earlier version of this study reported 66% for the description-based setup at L1. Table 12 crosses the old and new training code with the old and new split: the number follows the split and not the code. Independently, the ranking of setups depends on the recipe: under the screening recipe the example-based setup reaches only 75.7% on L1 against 94.5% for the description-based setup, with 20 epochs and cosine decay 85.6 against 98.5%, and under the paper recipe 95.4 against 98.5%. An undertrained recipe can therefore reverse the comparison, and the earlier symbolic failure was a property of that split.

Table 12: Where the earlier L1 result for the description-based setup came from: training code (old / new) by split (old / new), L1 exact match in percent, 2 seeds per cell except the top-left one (25 runs).

	old split	new split
old training code	65.8	95.0
new training code	66.5	98.5

C.6 L3 accuracy by rule family

Table 13 breaks L3 down by the family of the held-out instance on the L2-dial split. At the natural exposure level the description-based setup beats the example-based setup on value-shift and uniqueness rules and loses only on parity rules, which alone account for the aggregate gap. Per held-out world at that level, the ten worlds whose held-out instance has every factor pair supported by at least one trained rule score 87 to 100% under both setups; the twenty worlds with an unsupported pair score 21 to 94% (description-based) against 54 to 100% (example-based), the parity ones lowest (21 to 69% against 62 to 80%).

C.7 Further limitations of the evidence

- **One training recipe.** All setups share one recipe, and the ranking of setups is recipe-sensitive: under a shorter schedule the example-based setup is far from converged while the description-based

Table 13: L3 exact match in percent by rule family of the held-out instance, L2-dial split, 4 seeds, ten held-out worlds per family, at the lowest and the natural exposure level.

Exposure	Setup	value shift (links)	uniqueness	parity	aggregate
0%	description-based	75.8	66.3	34.2	58.7
0%	example-based	93.2	74.6	63.1	77.0
74.6%	description-based	98.3	83.8	51.2	77.8
74.6%	example-based	96.8	79.6	69.6	82.0

setup is not (Appendix C.5). The L0 and L1 comparison therefore holds under the training budget used; a budget sweep is left to future work. L0 is also measured on the held-out puzzles of the worlds used for checkpoint selection, so it is a mildly optimistic in-distribution reference.

- **Single-split dials and controls.** Both exposure dials, the description-dropout results, and most controls are run on one draw of one split. The below-floor result itself is replicated on six draws of two splits, but the shape of the dials and the size of the dropout repair are not. On the L2 dial, about 8 of the 31 points the description-based setup gains are accounted for by the floor rising with exposure.
- **Behavioural evidence only.** The account of the description as a crutch rests on accuracies, masked evaluations and the dropout repair; we did not analyse what the model represents, so it is consistent with the evidence rather than established by it.
- **One description alphabet.** The symbolic description uses one factored scheme (type, axis, position, parameter). Three ways of injecting it agree to within 2 points on L1 (Appendix C.4), so the injection mechanism is not the bottleneck, but no other alphabet or factorization was tried.

D Extended discussion

This appendix expands on how our parameterized-family framing relates to prior formal definitions of intelligence (extending Section 5).

Formal definitions of intelligence. Legg and Hutter (Legg & Hutter, 2007) and Hernández-Orallo (Hernández-Orallo, 2017) define intelligence as aggregate performance over a distribution of environments or tasks, weighted toward simpler ones, but give no concrete, computable task family with an explicit notion of difficulty. Our parameterized family of rule combinations is such a family: difficulty is set by how many rules, and which ones, are held out, and generalization within the family becomes a measurable quantity; a recent proposal similarly argues for measuring skill acquisition without human cognition as the reference point (Goldfeder et al., 2026). In Chollet's terms (Chollet, 2019), levels L0 to L2 sit within local generalization (new points from a known distribution), while L3 is a small step toward the broad generalization his spectrum places well beyond current systems.

E Architecture and training specification

This appendix completes Section 3.5: the full specification of the model and the training recipe. Appendix F gives the rationale behind task and data design choices the main text states without justification.

Figure 13 summarizes the overall flow described below.

Input. Tokens are embedded with a single embedding table shared by all setups, covering the 36 symbols of Section 3.2. The input is the conditioning prefix followed by the 36 cells of the puzzle: $16 + 36 = 52$ tokens for the description-based setup, $2 \cdot 36 + 36 = 108$ for the example-based setup, and 36 for the no-information baseline. Because the mixing weights are tied to absolute positions, no positional encoding is needed.

Recurrent core. Let x be the embedded input and (z_H, z_L) two latent states of the same shape, initialised randomly. One pass performs 6 updates of the fast state, each conditioned on the slow state

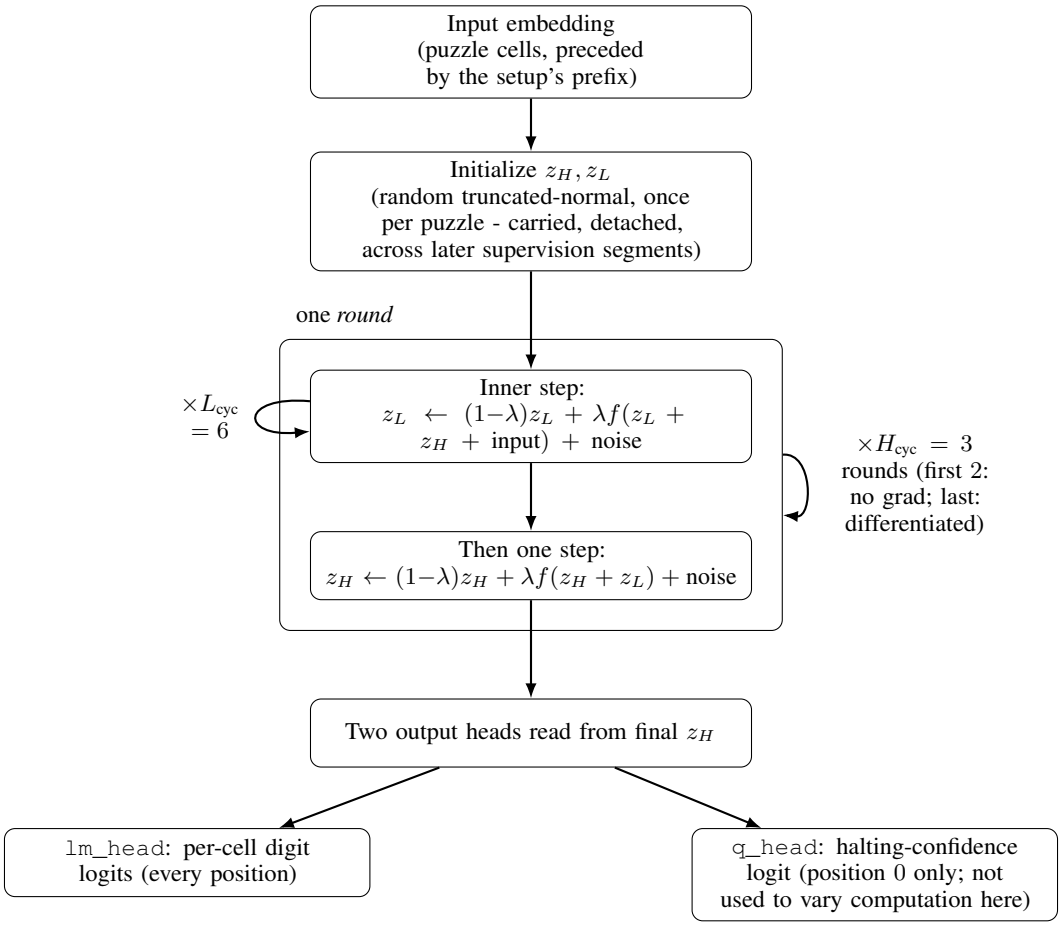


Figure 13: The reasoning engine, used unchanged across the three setups; only the input sequence (top box) differs between them. f is the 2-layer, 256-dimensional MLP-Mixer-style block stack described next, shared between the two update equations.

and the input, followed by one update of the slow state: $z_L \leftarrow f(z_L, z_H + x)$, then $z_H \leftarrow f(z_H, z_L)$, where f applies the stack of 2 blocks to the sum of its arguments. Updates are damped and noisy, $h \leftarrow (1-\lambda)h + \lambda\,\text{blocks}(h + \text{inj}) + \epsilon$ with $\lambda = 0.95$ and $\epsilon \sim \mathcal{N}(0, 0.01^2)$. Each forward call runs 3 such passes, of which only the last is differentiated; the earlier ones run without gradients.

Deep supervision and halting. Each training batch is processed in 8 consecutive segments. After every segment the loss is applied and an optimizer step is taken, and the latent states are carried to the next segment detached from the graph, so gradients never span segments. Two heads read the slow state: a linear head produces per-position logits over the vocabulary, of which only the last 36 positions are scored, and a scalar halting head reads the first position. The loss is a stablemax cross-entropy on the cells plus 0.5 times a binary cross-entropy that trains the halting head to predict whether the current prediction is exactly correct over all 36 cells. At evaluation the model is run for up to 32 rounds; a puzzle halts the first time the halting probability exceeds 0.9, and the prediction at that round is frozen and scored.

Size. The model has hidden size 256, 2 blocks, and expansion factor 4. The channel mixing, the embedding and the heads are the same everywhere; only the token-mixing MLPs depend on the sequence length, quadratically.

Optimization. AdamW, learning rate 10^{-4} with $\beta = (0.9, 0.95)$ and weight decay 0.01, 200 warmup steps followed by a constant rate, batch size 512, 500 batches per epoch and 20 epochs, which is $10{,}000$ batches, each with 8 optimizer steps (one per supervision segment), about five passes over the training puzzles. We keep the checkpoint with the best validation exact-match accuracy. The same recipe is used for every setup and every split; no per-setup tuning is performed. A shorter, decaying schedule leaves the example-based setup far from convergence while the description-based setup has converged, and reverses their ranking (Appendix C.5).

Deviations from HRM. Relative to Wang et al. (2025) we keep the hierarchical two-timescale recurrence, one-step-gradient training, and deep supervision, with three deviations: (i) attention blocks are replaced with MLP-Mixer-style (Tolstikhin et al., 2021) token/channel-mixing blocks, since the grid topology and the number of prefix slots are fixed and known in advance, so a fixed mixing pattern suffices; (ii) a single supervised halting head replaces HRM's Q-learning ACT mechanism, and the update counts above stay fixed during training, with halting used only at evaluation; (iii) plain AdamW replaces HRM's specialized optimizer.

F Design choices

This appendix gives the rationale behind task and data design choices the main text (Section 3) states without justification.

Why we chose 6×6 grids. Several pool rules tile the grid in 2×3 or 3×2 blocks, so the side length must be divisible by both 2 and 3; 6 is the smallest such size. Compared with the classic 9×9, it keeps a sweep over thousands of rule combinations tractable without making puzzles trivial: the no-information baseline reaches only 61% in-distribution exact-match accuracy on our main split, and at most 80% on any split we use. Nothing in the formalization of Section 2 is specific to this size.

Why the base is fixed. Holding the base fixed isolates the object of study: any generalization gap reflects the variable pool, not an entangled second source of novelty. The base also makes puzzle generation tractable. Its grid-wide coupling (every row and column is an all-different group) is what lets the solver of Appendix B propagate constraints and find uniquely completable partial configurations quickly, whereas most variable-pool rules are local (a box, a diagonal, one pair of rows) and constrain far fewer cells.

Why these rule types. Rules were chosen for their similarity to the base and to variants already familiar from Sudoku (boxes, diagonals, windows), each still introducing some difference, so the pool stays a single recognizable family rather than an arbitrary set. The pool is deliberately built so that most rules decompose into shared, reusable components (type, axis, position, parameter); the 30 link rules in particular cover every row or column pair at every offset k, so that each component appears in several contexts (Appendix A).

Padding control. Because the sequence length changes the size of the token-mixing MLPs and the compute per forward pass (Table 2), we additionally run the description-based setup with its prefix padded by null tokens to 108 positions, the example-based length, so that the two share the same token-mixing size and per-step compute, and the no-information setup padded to 52 positions, the description-based length. Table 10 reports both: padding the description worsens its L3 deficit on the axis split and removes most of it on the row-pair split, so sequence length is not a consistent explanation.