
THEIA: Learning Complete Kleene Three-Valued Logic in a Pure-Neural Modular Architecture

Neurosymbolic Artificial Intelligence
XX(X):2–26
©The Author(s) 2026
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/

Augustus Haoyang Li¹

Abstract

Many reasoning tasks have missing information. A system must pass “Unknown” through a chain of steps, and it must also see when a known value overrides it. Kleene’s strong three-valued logic (K3) gives exact rules for this. Earlier neural systems either build K3 gates into the network or give the logic to an external symbolic solver, so they do not show how a network can learn K3 from data. We present THEIA, a 2.75M-parameter modular network with four engines for arithmetic, order, set membership and logic. THEIA has no K3 gates and calls no solver. It learns all 39 K3 rules at more than 99% per-rule accuracy on 5 seeds. The engines have fixed boundaries, so we can compare what the network encodes at each boundary with what the input allows at that point. Linear and nonlinear probes show that “some input is Unknown” is decodable at every boundary, close to its Bayes ceiling. The verdict (the T/F/U output) stays below a 73.4% oracle reference until the Logic Engine, where the logic operator first enters. A Transformer also learns K3, but its verdict is already decodable from its input embeddings, so it does not allow this kind of check. On a mod-3 chain task with no absorbing states, a 1.51M-parameter THEIA step module trained on 5-step chains reaches 99.96% at 500 steps, and every seed stays at or above 99.9%. With the same training, flat MLPs fall to chance, a grid of residual MLPs rarely reaches 99%, and a pre-LN Transformer averages 99.24% with 3 of 5 seeds at or above 99%. Ablations show that the step architecture and end-to-end training explain this result, and the discretization method does not.

Keywords

Kleene three-valued logic, neurosymbolic reasoning, interpretability, probing, modular neural networks, length generalization

1 Introduction

Real reasoning often works with incomplete facts. A database row has a NULL field. A patient has a test result that is not back yet. A legal case has a fact that the court has not decided. In these settings, a system must carry “I don’t know” correctly through several steps. It must also know when one definite fact decides the answer anyway.

Kleene’s strong three-valued logic (K3) (Kleene 1952) gives exact rules for this. It adds a third value, Unknown (U), to True (T) and False (F). The values are ordered $F < U < T$. AND takes the minimum of its operands and OR takes the maximum, so every operator has a defined result when an operand is Unknown. In an *absorption rule* (also called a short-circuit rule), one definite operand fixes the result whatever the Unknown operand is, for example $F \wedge U = F$ and $T \vee U = T$. K3 is the logic behind SQL NULL handling, and it appears in reasoning with missing medical tests and with undecided legal facts.

Symbolic solvers such as Z3 (de Moura and Bjørner 2008) are formally correct, but they need hand-written rules and are not made for incomplete inputs. Neuro-symbolic systems (Manhaeve et al. 2021; Yang et al. 2020; Li et al. 2023) connect neural networks with symbolic solvers. In these systems the three-valued or probabilistic inference is done by the symbolic part, and the neural part gives perception or differentiable scores. Other neural approaches to K3 build K3 gates into the neurons (Chan et al. 1989; Wang and Shi 1998; Riegel et al. 2020; Damera et al. 2026). None of these systems can tell us how a network learns K3 from data, or where inside the network the three-valued computation happens.

We call a system *pure-neural* if all inference, including Unknown propagation, happens inside a network of generic MLP neurons. No neuron has a fixed K3 gate identity, and no external symbolic solver is used. A *gate-encoded* system instead builds K3 gate behavior into the architecture, so the logic is wired in and not learned.

In this paper we present THEIA*, a pure-neural modular network. It has four small MLP *engines* for arithmetic, order relations, set membership and logic. Each engine works on one domain, and the engines exchange 128-dimensional vectors along fixed edges. A sample asks a small compositional question. It gives integers a and b with an arithmetic operator (result c), an integer d with an order relation R , a set S , and a logic operator $\odot$. Any of a , b , d and S can be Unknown. The network must output the K3 verdict of “ $(c R d) \odot (c \in S)$ ”. For example, if d is Unknown, then $c R d$ is U . If $c \in S$ is False and $\odot$ is AND, the verdict is still F by absorption.

¹ Irvine Valley College, Irvine, CA, USA

Corresponding author:

Augustus Haoyang Li, Irvine Valley College, Irvine, CA, USA.
Email: hli132@ivc.edu

*Three-valued Hybrid Engine for Inference Architecture.

A Transformer can also learn K3. Why THEIA? This is the first question many readers ask, so we answer it first. A suitably trained 8-layer Transformer also learns all 39 K3 rules at more than 99% per-rule accuracy (Section 5), so learnability alone is not our claim. THEIA gives three things that the Transformer baseline does not give.

1. **Inspectable boundaries with known content.** Each engine output is a fixed *domain boundary*, and we know which inputs can reach it. This means we can compute, from the symbolic structure of the task, the best accuracy that any predictor could reach at that boundary, and we can test whether the network reaches it. In THEIA, the Unknown signal sits at or near this ceiling at every boundary, and the verdict appears only where the logic operator enters. In the Transformer, a linear probe already decodes the verdict from the input embeddings above our oracle reference, and its layers have no fixed meaning to check against.
2. **Reliable composition over long chains.** On a mod-3 chain task, every THEIA seed keeps at least 99.9% accuracy at 500 steps, $100\times$ the training length. No seed of the pre-LN Transformer reaches 99.9%, and 2 of its 5 seeds are below 99%.
3. **Faster training to the full rule set (auxiliary).** Under THEIA’s default optimizer settings, THEIA passes the targeted K3 diagnostic $6.5\times$ faster in wall-clock time. With Transformer-specific tuning, the gap narrows to about $3.6\times$.

The first point is the reason we see THEIA as a transparent-by-design neurosymbolic model. Its internal states can be checked against the symbolic structure of the task, and not only against its output labels.

Research questions. We study three questions. **RQ1:** Can a pure-neural network learn the complete K3 truth table, including all absorption rules in both operand orders, from data? **RQ2:** Where in the network are uncertainty and the verdict encoded, and does this match the data flow that the symbolic structure predicts? **RQ3:** Which step architectures keep K3-style composition reliable on chains $100\times$ longer than in training?

Contributions.

1. **Complete K3 learning without K3 primitives (RQ1).** THEIA passes all 39 K3 rules at more than 99% on all 5 seeds (195/195 rule–seed combinations, grand mean 99.88%). We also describe two silent errors in simple K3 test construction and a protocol that avoids them (Sections 4 and 5).
2. **A boundary analysis checked against symbolic ceilings (RQ2).** We derive the Bayes ceiling of the Has-Unknown signal at each boundary from the Unknown-flag structure, and we show that probes reach it. Verdict decodability stays below a 73.4% oracle reference at every upstream boundary, also for deeper MLP probes, and rises sharply at the Logic boundary. Controls on the non-Unknown subset, an operator-early variant and activation patching support this reading (Section 6).
3. **A reliability spectrum of step architectures (RQ3).** Under one shared training pipeline, flat MLPs fall to chance at 500 steps, a ResMLP grid reaches $\geq 99\%$ on only 3/20 trials, a pre-LN Transformer on 3/5 seeds, and THEIA on 5/5. An ablation shows that discretization is neither necessary nor sufficient for this result (Section 7).

Table 1. Main claims and where the evidence is.

Claim	Evidence	Section
THEIA learns all 39 K3 rules without K3 primitives	195/195 rule–seed combinations > 99%	5
Unknown is encoded at every boundary, near its Bayes ceiling	HU probe within 0.3pp of the ceiling at every upstream boundary	6.3
The verdict forms at the Logic Engine	Upstream verdict probes at most 70.17% (linear and MLP); 99.9% at Logic	6.2
The pattern is not only a result of operator placement	Operator-early variant keeps it	6.6
THEIA is the most reliable step module on 500-step chains	5/5 seeds $\geq$ 99.9%; Transformer 0/5; ResMLP 3/20 at $\geq$ 99%	7
Discretization does not explain the chain result	Soft, hard and fixed-temperature variants all $\geq$ 99.98%	7.3

4. **External checks.** On a multi-hop SQL-NULL task, THEIA keeps 100% accuracy and 0% unsafe verdicts up to 500 hops. Four-domain accuracy barely changes when the domain-to-engine map is broken (Sections 7.4 and 8).

Table 1 maps each claim to its evidence.

2 Related Work

Neuro-symbolic systems. DeepProbLog (Manhaeve et al. 2021) adds neural predicates to probabilistic logic programming. NeurASP (Yang et al. 2020) combines neural networks with Answer Set Programming, and Scallop (Li et al. 2023) gives differentiable reasoning over Datalog. All three leave the three-valued or probabilistic inference to an external symbolic component (Marra et al. 2024). Logic Tensor Networks (Serafini and d’Avila Garcez 2016; Badreddine et al. 2022), Neural Logic Machines (Dong et al. 2019) and semantic-loss methods (Xu et al. 2018) keep inference inside the network, but they use two-valued or fuzzy semantics and not K3. Neural module networks (Andreas et al. 2016) were an early modular architecture for compositional reasoning, and they inspired the engine design of THEIA.

Three-valued logic in neural networks. K3 is the basis of SQL NULL handling and of some semantics for logic programs (Fitting 1985). Earlier neural approaches build K3 gate primitives into the architecture at different levels. The Chan–Hsu–Teh family of neural logic networks (Chan et al. 1989; Hsu et al. 1991; Teh 1995) encodes each truth value as an ordered pair of neuron activations, and the three-valued connectives come from fixed combinations of neurons. Hsu et al. (1990) extend this to multi-valued networks with hard-coded gate semantics. TMLNN (Wang and Shi 1998) trains multi-valued neuron parameters but keeps hand-engineered three-valued gates. Logical Neural Networks (LNN) (Riegel et al. 2020) give each neuron a logical connective and run symbolic inference at runtime. LNN needs a hand-specified formula structure and does

not handle numerical functions, so it cannot evaluate $a + b$, $c > d$ or $c \in S$, and we cannot use it as a baseline here.

Differentiable logic gate networks (Petersen et al. 2022, 2024) learn compact Boolean circuits, where each neuron learns a distribution over the 16 two-input Boolean gates. At the same time as our work, Damera et al. (2026) extend them to K3 (DTLGN). Each neuron picks one of the 19,683 two-input K3 gates, trained through a polynomial surrogate. DTLGN and THEIA share the goal of learning K3 neurally, and neither calls a solver at runtime. The difference is where the K3 primitives live. In DTLGN, each neuron’s gate identity comes from a fixed K3 gate set. In THEIA, neurons are generic MLP units, and the K3 algebra appears at the representation level, through prototype matching in a learned 128-dimensional space. As far as we know, no earlier system shows that a neural network learns the complete K3 truth table, with the absorption rules in both operand orders, under the pure-neural criterion. Pretrained language models that apply K3 rules in context are a different setting, because they bring logical priors from their training text.

Neural algorithmic reasoning and length generalization. CLRS (Veličković et al. 2022) studies deterministic two-valued algorithmic reasoning. Discrete neural algorithmic reasoning (DNAR) (Rodionov and Prokhorenkova 2025) is the closest mechanism to our chain experiment. It also forces discrete states between steps, but it uses a specialized processor on two-valued algorithms. THEIA uses a generic MLP step module, rounds the chain state to one-hot with Gumbel-softmax, and trains the chain end to end. Our mod-3 chain is a state-tracking task, a running sum in $\mathbb{Z}_3$. Such tasks separate architectures under length generalization (Delétang et al. 2023; Liu et al. 2023; Csordás et al. 2022). The experiment is also close to studies of length generalization in Transformers (Dziri et al. 2023; Press et al. 2022; Anil et al. 2022) and to SCAN (Lake and Baroni 2018). Our graph results relate to work on what message-passing networks can compute (Xu et al. 2020; Loukas 2020) and on over-smoothing (Li et al. 2018; Chen et al. 2020).

Probing and mechanistic interpretability. Our boundary analysis extends linear probing (Alain and Bengio 2017) and belongs to the mechanistic-interpretability line of work (Elhage et al. 2021; Nanda et al. 2023). We also use activation patching, a form of causal mediation analysis on internal activations (Vig et al. 2020; Geiger et al. 2021; Meng et al. 2022). A common problem in probing is that we do not know how much information a layer *could* carry. In THEIA, the fixed data flow lets us compute this bound exactly for the Unknown signal.

3 The THEIA Architecture

3.1 Task

Each sample gives integers a and b and an arithmetic operator $\oplus$, with result $c = a \oplus b$. It also gives an integer d with an order relation R , a set S as a 21-dimensional binary vector, and a logic operator $\odot$ drawn uniformly from $\{\wedge, \vee, \neg, \rightarrow, \leftrightarrow\}$. Each of a , b , d and S is independently replaced by a learned Unknown embedding with probability $P_{\text{unk}} = 0.15$.

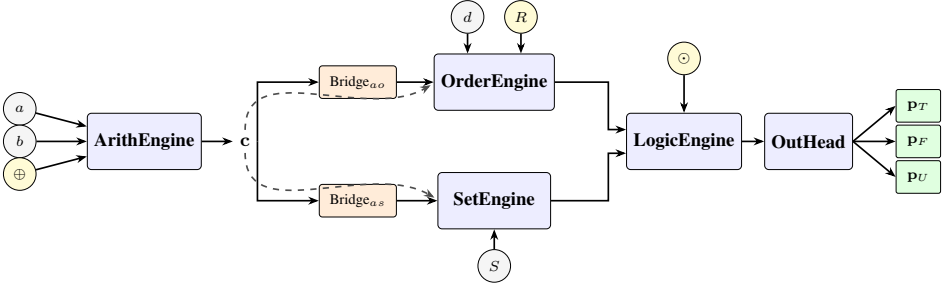


Figure 1. THEIA architecture. Four engines handle four separate domains. They share no parameters, and there is no attention between domains. ARITHENGINE maps the integer operands a , b and the arithmetic operator $\oplus$ to c , a 128-dimensional vector for $a \oplus b$. Two *bridges* (orange), small residual MLPs, carry c to ORDERENGINE and SETENGINE. The dashed arrows are the residual bypass, so c is added to the bridge output. ORDERENGINE tests the order relation R between c and the integer d . SETENGINE tests whether c is in the set S , given as a 21-dimensional binary vector. ARITHENGINE runs first, and then ORDERENGINE and SETENGINE run in parallel. LOGICENGINE applies the logic operator $\odot$ to their two outputs. OUTHEAD maps the result to a vector α , and the verdict is the prototype (p_T , p_F or p_U) with the highest cosine similarity to α .

The truth value of $c R d$ is the *Order value*, and the truth value of $c \in S$ is the *Set value*. The target is the K3 value that $\odot$ gives for these two values. The unary $\neg$ uses only the Order value.

We use the K3 semantics of Section 1 ($\wedge$ is min, $\vee$ is max) with the encoding $F=0$, $U=\frac{1}{2}$ and $T=1$. Implication and biconditional are defined as

$$\begin{aligned} a \rightarrow b &\equiv \neg a \vee b, \\ a \leftrightarrow b &\equiv (a \rightarrow b) \wedge (b \rightarrow a), \end{aligned}$$

and negation swaps T and F and keeps $\neg U = U$. These definitions give every entry of the truth table, including all absorption rules. For example, $F \rightarrow U = \neg F \vee U = T$, and $T \leftrightarrow U = (T \rightarrow U) \wedge (U \rightarrow T) = U$. The five operators give 39 rules in total: 36 binary rules and 3 NOT rules. Because the truth table is small, we can test every rule, and we can also look for where in the network the verdict forms.

3.2 Engines and data flow

THEIA processes the question in a fixed sequential–parallel topology (Figure 1). We call this design *domain-separated encoding*. Each domain has its own engine, with no shared parameters and no attention across domains. Information moves between domains only

through the bridges. The forward pass is:

$$\mathbf{c} = \text{ArithEngine}(a, b, \oplus) \quad (1)$$

$$\mathbf{v}_{\text{ord}} = \text{OrderEngine}(\text{Bridge}_{ao}(\mathbf{c}) + \mathbf{c}, d, R) \quad (2)$$

$$\mathbf{v}_{\text{set}} = \text{SetEngine}(\text{Bridge}_{as}(\mathbf{c}) + \mathbf{c}, S) \quad (3)$$

$$\mathbf{o} = \text{OutHead}(\text{LogicEngine}(\mathbf{v}_{\text{ord}}, \mathbf{v}_{\text{set}}, \odot)) \quad (4)$$

Here $\mathbf{o}$ is the final output, and the two bridges are residual MLPs (Linear $\rightarrow$ GELU $\rightarrow$ LayerNorm) with 33K parameters in total. As in message-passing neural networks (Gilmer et al. 2017), the engines exchange vectors along fixed edges. Eq. 4 is important for the analysis later. The Logic Engine receives only the Order output, the Set output and the logic operator. There is no residual path from $\mathbf{c}$ or from the input Unknown flags to the Logic Engine.

Each engine is a small MLP stack. The Arithmetic, Order, Set and Logic engines have 0.40M, 1.18M, 0.21M and 0.91M parameters. With the bridges and the output head with prototypes (17K), the total is 2.75M. The Arithmetic Engine fuses a number encoder with an operator embedding. The Order and Logic engines each use three parallel sub-MLPs, and each pair of sub-MLP outputs is fused. These are the Global, Local and Event subspaces in the Order Engine and the Conjunctive, Disjunctive and Implicative subspaces in the Logic Engine. The Set Engine has its own Unknown embedding. The split into subspaces helps convergence, but the model is also correct without it (Section 5.1).

3.3 Output head

The output uses three prototype vectors $\{\mathbf{p}_T, \mathbf{p}_F, \mathbf{p}_U\} \in \mathbb{R}^{128}$, one for each verdict, as in prototypical networks (Snell et al. 2017). They are initialized to be mutually orthogonal and serve as class anchors. The logit for verdict v is the dot product $\mathbf{o} \cdot \mathbf{p}_v$. At inference, the verdict is the prototype with the highest cosine similarity to $\mathbf{o}$. The two rules pick the same verdict whenever the three prototypes have equal norm. We train with cross-entropy loss, and the Unknown class gets twice the loss weight of the other classes ($w_U = 2.0$). The prototypes are learned, and they move away from orthogonality during training. The prototype head is standard metric-learning practice and contains no K3 rule. All baselines in this paper use the same kind of 3-class prototype head.

3.4 Why these four domains

The four domains cover four kinds of indeterminacy for K3: numerical, relational, categorical and propositional. Each maps an Unknown input to its output in a different way. An Unknown operand in $a + b$ gives an unknown number. An Unknown d in $c > d$ gives an unknown truth value. An Unknown c in $c \in S$ gives an unknown membership. In the logic domain, two operands that may be Unknown are combined by a K3 operator, and the absorption rules apply. Within propositional K3 over arithmetic inputs, none of these can stand in for another. The set is minimal, but it does not cover every logic. We do not test first-order logic or other many-valued logics (Section 8.4).

4 Experimental Setup

Data and training. All experiments run on one NVIDIA RTX 5080 GPU (16GB). We use AdamW (Loshchilov and Hutter 2019) ($\text{lr} = 10^{-3}$, weight decay 0.01) with cosine annealing and FP16 mixed precision. The four-domain experiments use 2M samples with an 80/20 train-test split, so the test set has 400K samples. The integer inputs go up to 20, and $P_{\text{unk}} = 0.15$. On the $N=50,000$ probe sample, about 27% of verdicts are False, 33% True and 40% Unknown. The multi-seed experiments use the seeds {42, 123, 256, 777, 999}.

Per-rule K3 diagnostic. For each of the 39 rules we build 10,000 test samples whose Order and Set values are fixed to the operands of that rule. We found two construction errors that silently corrupt such a test, and they affected some of our earlier runs. First, a simple way to make the Order value Unknown is to set a to Unknown. But a also feeds c , and c feeds the Set Engine, so the Set value also becomes Unknown. The test then checks (U, U) in place of the intended $(U, \text{definite})$, and a different K3 rule decides the verdict. We therefore inject Unknown through d and keep a and b definite. Second, to force a True Order value under “strictly greater than”, one may set $d = \max(0, c - 1)$. When $c = 0$ (about 5% of samples), this gives $0 > 0$, which is False. The per-rule accuracy then still looks plausible, but it is biased downward. We use “greater than or equal” with the same d , which is always True. All reported diagnostics use both fixes. Supplementary Section B gives the details. The reported per-rule results used hash-based data seeds, and the released diagnostic uses deterministic per-rule seeds, so re-computed per-rule accuracies can differ by up to about 0.2 percentage points (pp). No rule changes its $> 99\%$ result.

Why per-rule accuracy is the main metric. The six absorption rules ($F \wedge U, U \wedge F, T \vee U, U \vee T, F \rightarrow U, U \rightarrow T$) are among the rarest rules in the training data. They belong to the 16 rules with one definite and one Unknown input, and each of these is only about 0.92% of the training samples (Supplementary Section B). A model can thus reach high overall accuracy while rare rules are still unreliable. We therefore use a *Kleene-aware stopping criterion* in all K3 experiments. Training stops when the overall accuracy is above 99.9% and all 12 targeted rules (Table 2) are above 99%, on two consecutive checkpoints.

Baselines. We use two Transformer variants (Vaswani et al. 2017), each with 8 layers, 8 heads, model width 192 and 3.58–3.64M parameters. Post-LN and pre-LN mean that layer normalization comes after or before each sublayer. We use the post-LN Transformer for the K3 and probing comparisons. It has the extra seeds {31415, 27182, 14142}, for 8 seeds in total. For the chain pipeline (Section 7) we use the pre-LN Transformer (Xiong et al. 2020), because the post-LN Transformer does not converge under the shared learning rate in the three-phase chain training. The chain pipeline also uses flat MLPs and a grid of residual MLPs (ResMLPs) as step modules. *Plateau-restart*, which reinitializes a model when accuracy stalls, is disabled for both architectures in the four-domain task and enabled for all architectures in the chain pipeline.

Table 2. Per-rule K3 diagnostic on the 12 targeted rules. Each has one Unknown operand. They are the eight $\wedge/\vee$ rules (Unknown in either position) and the four $\rightarrow/\leftrightarrow$ rules with Unknown second. Mean $\pm$ std over 5 seeds, with 10,000 samples per rule per seed. “Min” is the lowest single-seed accuracy. $\dagger$: Unknown is the first operand. All 60 rule–seed combinations are above 99%.

Expression	Expected	Mean $\pm$ Std (%)	Min (%)
$F \wedge U$	F	99.97 ± 0.06	99.85
$T \wedge U$	U	99.94 ± 0.04	99.89
$U \wedge F^\dagger$	F	99.93 ± 0.06	99.84
$U \wedge T^\dagger$	U	99.80 ± 0.11	99.69
$T \vee U$	T	99.93 ± 0.08	99.77
$F \vee U$	U	99.92 ± 0.11	99.71
$U \vee T^\dagger$	T	99.93 ± 0.06	99.85
$U \vee F^\dagger$	U	99.85 ± 0.10	99.68
$F \rightarrow U$	T	99.95 ± 0.03	99.91
$T \rightarrow U$	U	100.00 ± 0.00	99.99
$T \leftrightarrow U$	U	100.00 ± 0.00	100.00
$F \leftrightarrow U$	U	100.00 ± 0.00	100.00
Grand mean (12/12, 5/5 seeds)		99.94	–

Hypotheses. We state the three main claims as falsifiable hypotheses before the results. H1: in the discretized mod-3 chain task, flat MLP and ResMLP step modules do not keep $\geq 99\%$ at 500 steps on a majority of seeds, while THEIA does. H2: at every upstream boundary, the Has-Unknown signal is decodable at $\geq 75\%$, and verdict decodability stays at or below the 73.4% oracle reference (Section 6). H3: under shared optimizer settings, THEIA reaches 12/12 targeted rules in less wall-clock time than the 8-layer Transformer. Supplementary Section A gives the full falsification criteria and lists the analysis choices that we fixed after seeing the data.

5 Learning the Complete K3 Truth Table (RQ1)

Table 2 shows the 12 targeted rules. All pass at more than 99% on all 5 seeds (60/60 rule–seed combinations). The AND and OR absorption rules hold in both operand orders. With Unknown first ($U \wedge F^\dagger$, $U \vee T^\dagger$), both have a mean of 99.93%. The sixth absorption rule, $U \rightarrow T$, is not part of the stopping criterion, and it still reaches $99.93 \pm 0.07\%$ on all 5 seeds.

Table 3 covers the other 27 rules. The three NOT rules are trained and tested directly with the unary operator. We do not build them from the identity $\neg x \equiv x \rightarrow F$, so they differ slightly from the $F \rightarrow F$, $T \rightarrow F$ and $U \rightarrow F$ entries. On the full 39-rule table, 195/195 rule–seed combinations pass at more than 99%. The grand mean is 99.88%, and the worst single combination is $F \vee F$ at seed 999, with 99.15%. The lowest rule mean is also a classical rule, $F \vee F = F$ at 99.54%. We think the reason is the training distribution. The (F, F) input to the final disjunction needs two independent upstream domains to both be False, so it is only about 2.61% of training samples. The diagnostic

Table 3. Full K3 diagnostic for the other 27 rules. Mean $\pm$ std over 5 seeds, with 10,000 samples per rule per seed. All 27 rules are above 99% on all 5 seeds (135/135 rule–seed combinations).

Rule	Acc (%)	Rule	Acc (%)	Rule	Acc (%)
<i>Definite–definite (18 rules including NOT)</i>					
$F \wedge F$	100.00 ± 0.01	$F \vee F$	99.54 ± 0.27	$F \rightarrow F$	99.74 ± 0.12
$F \wedge T$	99.84 ± 0.05	$F \vee T$	99.92 ± 0.06	$F \rightarrow T$	99.98 ± 0.03
$T \wedge F$	99.93 ± 0.08	$T \vee F$	99.96 ± 0.03	$T \rightarrow F$	99.86 ± 0.12
$T \wedge T$	99.83 ± 0.09	$T \vee T$	100.00 ± 0.00	$T \rightarrow T$	99.86 ± 0.07
$F \leftrightarrow F$	99.74 ± 0.12	$F \leftrightarrow T$	99.66 ± 0.15	$\neg F$	99.72 ± 0.13
$T \leftrightarrow F$	99.88 ± 0.11	$T \leftrightarrow T$	99.81 ± 0.10	$\neg T$	99.96 ± 0.03
<i>Unknown–involving (9 rules not in Table 2)</i>					
$U \wedge U$	100.00 ± 0.00	$U \vee U$	100.00 ± 0.00	$U \rightarrow F$	99.89 ± 0.10
$U \rightarrow T$	99.93 ± 0.07	$U \rightarrow U$	100.00 ± 0.00	$U \leftrightarrow F$	100.00 ± 0.00
$U \leftrightarrow T$	100.00 ± 0.00	$U \leftrightarrow U$	100.00 ± 0.00	$\neg U$	100.00 ± 0.00

tests this input alone with 10,000 samples, so a mild distribution shift is the most likely cause, not the difficulty of the rule.

The Transformer also learns K3. With a standard Transformer training recipe (Section 5.2), the post-LN Transformer also passes all 39 rules at more than 99% on three seeds (117/117 rule–seed combinations, grand mean 99.89%). Its worst rules are $F \leftrightarrow T$ on seed 42 (99.48%) and $T \rightarrow F$ on seeds 123 and 256 (both 99.50%). None of these has an Unknown operand. We use the same ground truth, coverage and pass threshold for both models. The answer to RQ1 is therefore yes, both for THEIA and for a tuned Transformer. The rest of the paper studies what differs between the two.

5.1 Component ablations

Each ablation below is a single training run on the four-domain task, where accuracy is near ceiling, so these runs test whether a component is needed for correctness, not how large its effect is. Supplementary Section C gives the full results.

- **Logic subspaces.** Replacing the three Logic subspaces with a single MLP (2.49M parameters in total) still passes all 12 targeted rules at $> 99\%$. The worst rule is $F \vee U$ at 99.73%.
- **Bridges.** Removing the bridges (33K parameters, 1.2% of the total) gives a 2.72M model that also passes 12/12. The worst rule is $T \wedge U$ at 99.94%.
- **Prototype initialization.** Random initialization from $\mathcal{N}(0, 0.02)$ on seed 42 keeps 12/12, with all rules $\geq 99.89\%$. Wall-clock time grows by $\sim 15\%$ (10.68 vs. 9.24 min), where 9.24 min is the seed-42 run in the 5-seed mean of Section 5.2. Orthogonal initialization helps speed but is not needed for correctness.

Table 4. Four-domain comparison under the matched optimizer settings, which are THEIA’s development default. THEIA uses 5 seeds. The Transformer uses 8 seeds, and its time is over the 7 converged seeds ($\dagger$). Both models pass all 12 targeted rules at $> 99\%$ on every converged seed.

Model	Accuracy	Params	Seeds 12/12	Time (min)
THEIA (modular)	$99.96\% \pm 0.01\%$	2.75M	5/5	7.93 ± 1.40
Transformer (8L, 8H)	$99.98\% \pm 0.01\%$	3.64M	7/8	$51.5 \pm 11.0^\dagger$
<i>THEIA speedup</i>				$6.5\times$

- **Class weights.** With uniform class weights the model also keeps 12/12, and the U-class accuracy is the same (99.98% in both settings). Uniform weights also converged faster in this run (6.73 vs. 10.53 min for the default weights, timed in a separate paired run). The Unknown upweighting is not needed.
- **Shift in the Unknown rate.** A model trained at $P_{\text{unk}}=0.05$ and tested at $P_{\text{unk}}=0.50$ (a $10\times$ shift) passes all 12 targeted rules at $> 99\%$, with overall accuracy 99.9975%. This shows that the model does not only fit the Unknown rate seen in training.

5.2 Training time (auxiliary)

We measure the wall-clock time to pass all 12 targeted rules. Under *matched optimizer settings*, both models use one AdamW configuration (lr 10^{-3} , cosine schedule, batch 4096, no warmup), which is THEIA’s development default. THEIA (2.75M parameters) reaches 12/12 on 5/5 seeds in 7.93 ± 1.40 min. The 8-layer post-LN Transformer (3.64M parameters) reaches 12/12 on 7 of 8 seeds in 51.5 ± 11.0 min (Table 4). Its failed seed (123) converges under tuning, which points to optimizer sensitivity. The gap is $6.5\times$, or $4.9\times$ per parameter.

We chose the thresholds of the Kleene-aware criterion by looking at THEIA’s training curves, so we also check a criterion that uses only overall accuracy ($\geq 99.9\%$). On the initial 5 seeds, THEIA needs 5.7 ± 1.4 min and the Transformer 39.5 ± 7.1 min ($n=4$), a gap of $7.0\times$. At looser thresholds the gap grows, to $9.8\times$ at $\geq 99.5\%$ and $12.2\times$ at $\geq 99.0\%$.

Because the matched settings are not the usual Transformer recipe, we also train the Transformer with a standard recipe: peak lr 10^{-4} , AdamW $\beta = (0.9, 0.98)$, 5-epoch linear warmup then cosine, weight decay 0.01 and batch 2048 ($n=3$). The tuned Transformer first passes all 12 rules about $3.6\times$ later than THEIA. At that point its overall accuracy is still below 99.9% on all 3 seeds, so this ratio compares its first 12/12 crossing with THEIA’s full criterion. We also applied the same recipe to THEIA itself. With it, THEIA reaches 12/12 on all 5 seeds in 9.88 ± 1.42 min, about 25% slower than with its default settings. Supplementary Section G gives the per-seed details.

6 Where Uncertainty and the Verdict Are Encoded (RQ2)

This section asks what each engine output contains. The fixed data flow of Eqs. 1–4 makes this question precise. The Unknown flags of a , b , d and S all enter before the Logic Engine, but the logic operator enters only at the Logic Engine. We can therefore compute what information is available at each boundary, and we can test whether the network actually encodes it.

6.1 Probes and references

We take the output vector of each engine, called a *domain boundary*. The boundaries before the Logic Engine (Arithmetic, Order, Set) are *upstream*. At each boundary we train *probes*, which are linear SVM or small MLP classifiers, and we call their test accuracy *decodability*. All analyses use the same 5 checkpoints as Table 2, with 50K samples per checkpoint drawn with an independent data seed.

We use two references. The *Has-Unknown signal* (HU) is 1 if any of a , b , d and S is Unknown. Its majority baseline is about 52%. The four Unknown flags a_u , b_u , d_u and s_u are i.i.d. Bernoulli(0.15). A flag is *visible* at a boundary if its input reaches that engine or an engine upstream of it. If k flags are visible, the best possible predictor of HU marginalizes over the hidden ones, and its accuracy, the *Bayes ceiling*, is

$$\text{Ceiling}(k) = (1 - 0.85^k) + 0.85^k \cdot 0.85^{4-k}.$$

The Arithmetic boundary sees a_u and b_u ($k=2$), so its ceiling is 0.7995. The Order boundary sees a_u , b_u , d_u , and the Set boundary sees a_u , b_u , s_u ($k=3$). Their ceiling is 0.9079. The Logic boundary sees all four flags, and its ceiling is 1.0.

For the verdict, we use the *oracle reference*. It is the accuracy of an ideal predictor that knows only whether the true verdict is U. It labels all U samples correctly and guesses the majority class for the rest. A fraction 0.4027 of the $N=50,000$ samples have verdict U, and among the other samples the majority class has a share of 0.5550, which gives a reference of $0.4027 \times 1.0 + 0.5973 \times 0.5550 = 0.734$. This is a benchmark and not the Bayes ceiling of a boundary, because it uses the full input. Knowing whether the verdict is U is also more than knowing HU, because an absorption rule can make the verdict definite when an input is Unknown.

Table 5 gives the linear probes for six variables. Three things are clear. First, the logic operator is not decodable before the Logic Engine (at most 20.2%, chance 20%). This is expected from the data flow, and it also shows that training does not create another path for the operator. Second, each engine’s own variable is decodable at its own boundary: the Order value at Order (1.000) and the Set value at Set (0.981). The Order and Set values are decodable at about 58% at the other upstream boundaries, because c reaches both engines. Third, the arithmetic result keeps $R^2 \approx 0.83$ at the Arithmetic and Order boundaries, drops to 0.81 at Set, and falls to 0.16 at Logic.

6.2 The verdict side

The linear 3-class verdict probe reaches 60.9%, 67.2% and 69.7% at the Arithmetic, Order and Set boundaries. Then it rises sharply to 99.9% at the Logic boundary. All

Table 5. Linear probes at the domain boundaries. Each probe decodes one variable from a 128-dimensional boundary vector. Mean $\pm$ std over 5 checkpoints, with 50K samples per checkpoint. Chance is 33% for a truth value and 20% for the logic operator. The arithmetic result is measured by R^2 . An entry of 0.000 means std $< 5 \times 10^{-4}$. This table uses the original 70/30 split with the epoch chosen on the test split. Table 6 uses a stricter protocol.

Probe	Arith	Order	Set	Logic
Arith result (R^2)	0.837 ± 0.010	0.837 ± 0.021	0.811 ± 0.013	0.156 ± 0.062
Order TV (acc)	0.583 ± 0.002	1.000 ± 0.000	0.584 ± 0.001	0.973 ± 0.015
Set TV (acc)	0.588 ± 0.003	0.585 ± 0.003	0.981 ± 0.009	0.995 ± 0.003
Final verdict (acc)	0.609 ± 0.001	0.672 ± 0.002	0.697 ± 0.002	0.999 ± 0.000
Logic op (acc)	0.199 ± 0.002	0.198 ± 0.002	0.202 ± 0.004	0.884 ± 0.041
Has Unknown (acc)	0.802 ± 0.000	0.911 ± 0.000	0.908 ± 0.000	0.997 ± 0.003

Table 6. Verdict decodability for the linear SVM probe and MLP probes of depth $d \in \{2, 4, 6\}$ (hidden width 256). All probes use a random 60/20/20 split, with the MLP epoch and the SVM C chosen on validation and one test evaluation. Mean $\pm$ sample std over 5 seeds. Every upstream cell is below the 73.4% oracle reference.

Boundary	Linear SVM	MLP $d=2$	MLP $d=4$	MLP $d=6$
Arith	0.611 ± 0.001	0.612 ± 0.001	0.612 ± 0.000	0.613 ± 0.000
Order	0.672 ± 0.002	0.672 ± 0.000	0.672 ± 0.002	0.672 ± 0.001
Set	0.693 ± 0.001	0.698 ± 0.002	0.699 ± 0.001	0.699 ± 0.001
Logic	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000

upstream values are below the 73.4% oracle reference. In other words, no upstream probe does better than an oracle that knows only whether the verdict is U.

A linear probe may miss information that a nonlinear probe can read, so we also train MLP probes with depth $D \in \{2, 4, 6\}$ and hidden width 256. They use a stricter 60/20/20 train/validation/test split. We choose the epoch (and the SVM C) on the validation split and evaluate the test set only once. Table 6 gives the results. At $D=2$, the MLP probe reaches 61.2%, 67.2%, 69.8% and 99.9% at the four boundaries. The highest upstream 5-seed mean over all depths is $69.92 \pm 0.12\%$ (Set, $D=6$), so all upstream means stay below 70.0%. The highest single value is 70.09% (Set, $D=4$, seed 999). Making the probe three times deeper changes the upstream values by at most +0.12pp. A separate check with hidden width 512 gives at most $70.17 \pm 0.21\%$ upstream (Supplementary Section D). The verdict is therefore not hidden in a nonlinear form at the upstream boundaries, at least for the probe families we tested.

Part of this result comes from the data flow. The verdict depends on the operator, and the operator enters only at the Logic Engine, so the upstream verdict probes are limited partly by the architecture. We do not read the result as the network “choosing” to delay its verdict. What the result shows is that the network does not find a way around the data flow. It does not carry more verdict information upstream than an oracle that knows only the U/non-U split.

Table 7. Bayes ceilings of HU decoding at each boundary and the measured probe accuracy. *Visible*: the Unknown flags that reach the boundary. *Ceiling*: closed form / empirical (majority-class accuracy over the observed visible-flag patterns, $N=50,000$). *70/30*: the original protocol, 5-seed mean. $w=512$: a probe with hidden width 512, a stratified 60/20/20 split and validation-based epoch selection (5-seed mean). *Gap*: $w=512$ accuracy minus the empirical ceiling.

Boundary	Visible	Ceiling (closed / empirical)	70/30	$w=512$	Gap
Arith	$\{a_u, b_u\}$	79.95% / 79.98%	80.23%	80.03%	+0.05pp
Order	$\{a_u, b_u, d_u\}$	90.79% / 90.85%	91.06%	91.09%	+0.24pp
Set	$\{a_u, b_u, s_u\}$	90.79% / 90.74%	90.75%	90.77%	+0.03pp
Logic	all four	100.00% / 100.00%	$99.65 \pm 0.33\%$	$99.76 \pm 0.22\%$	-0.24pp

6.3 The uncertainty side

The HU probe reaches 80.0% at the Arithmetic boundary, 91.1% at Order, 90.8% at Set and 99.8% at Logic. These are 5-seed means for a probe with hidden width 512 and the validation-based protocol. The cross-seed std is at most 0.01pp upstream and 0.22pp at Logic. Table 7 compares these values with the Bayes ceilings. At the Arithmetic and Set boundaries the probe is within 0.05pp of the empirical ceiling. The network keeps almost all Unknown information that the visible flags carry.

The Order value is 0.24pp above its empirical ceiling. This cannot be information about s_u . By Eq. 2, $\mathbf{v}_{\text{ord}}$ depends only on $(a, b, \oplus, d, R)$, which are independent of s_u , and the empirical flag marginals are within 0.003 of 0.15. It is a finite-sample effect. The ceiling is estimated on all 50,000 samples, but the probe is scored on a test split of 10,000 samples, where the standard error of the Order ceiling is about 0.26pp. The Logic boundary is slightly below its ceiling of 100%. An exact lookup over the four flags would reach it. We read this shortfall as the usual approximation error of an MLP on a Boolean function of a few bits.

Together, the two sides give the pattern we call *uncertainty-verdict asymmetric propagation*. HU is decodable at every boundary, at or near the ceiling that the data flow allows. The verdict stays below the oracle reference upstream and appears only at the Logic boundary.

6.4 Class geometry

Figure 2 shows the same pattern without a probe. We use the cosine between class centroids because it does not change under rescaling. At the Arithmetic boundary, the F and T centroids are nearly colinear. This fits the fact that the verdict also depends on operands that enter at later engines. The F-T cosine drops at the two parallel boundaries (Order, Set), and it is lowest at the Logic boundary, where the three centroids point further apart than orthogonal vectors. At Order, the F-U and T-U cosines are near zero on average but vary widely across seeds, so only the F-T trend is stable there. Supplementary Section D gives the Euclidean distances (the F-T distance grows by $1,898\times$ from Arithmetic to Logic) and a t-SNE view.

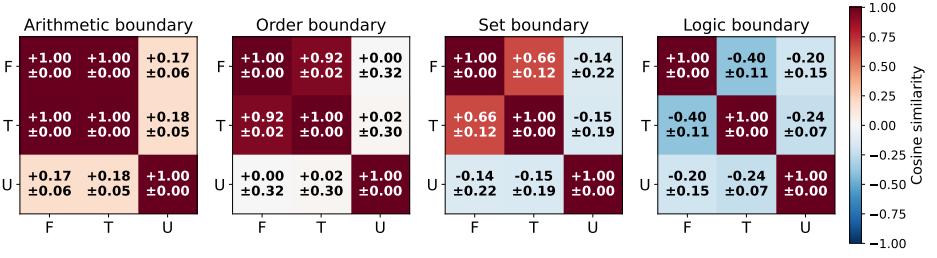


Figure 2. Cosine similarity between class centroids (F, T, U) at each domain boundary, mean $\pm$ std over 5 checkpoints. At the Arithmetic boundary, the F–T cosine is $+0.999 \pm 0.001$, so the F and T centroids point in almost the same direction. The F–U and T–U cosines there are $+0.175 \pm 0.055$ and $+0.178 \pm 0.053$, so Unknown is already partly separated. The F–T cosine falls from $+0.999$ at Arithmetic to $+0.923$ at Order and $+0.663$ at Set, and then to -0.402 at Logic, where all off-diagonal cosines are negative.

Table 8. Operator-identity probe at the three upstream boundaries. A 5-class linear SVM predicts the logic operator from the boundary vector (5 seeds, NU subset, 50K samples, 70/30 split). All three boundaries are at the 20% chance level.

Boundary	Probe acc (%)	Δ vs. chance	Result
Arith	19.88 ± 0.16	-0.12pp	at chance
Order	19.78 ± 0.25	-0.22pp	at chance
Set	20.25 ± 0.38	$+0.25\text{pp}$	at chance

6.5 The non-Unknown subset

One result could suggest that the verdict is partly decided before the Logic Engine. On the *NU subset*, the samples whose true verdict is T or F, the Set-boundary verdict probe reaches 60.13%. This is $+4.09\text{pp}$ above the *mixed-operator NU majority* (56.04%), the majority-class accuracy on this subset with all operators pooled. The oracle reference does not apply here, because every NU sample has a non-U verdict. We examine the lift with three controls (70/30 split, linear SVM).

First, an *operator-identity probe* tries to predict the logic operator from each upstream boundary (Table 8). All three boundaries are at chance, which means no upstream boundary linearly encodes the operator.

Second, a *per-operator stratified probe* is the Set-boundary verdict probe trained separately within each operator’s NU samples (Table 9). For AND, OR and IMP, its accuracy is 6–7pp below that operator’s NU majority. For IFF both are near 50%. Verdict leakage would give the opposite result.

Third, the lift is explained by the Set value itself. The Set Engine is built to output val_{set} , the truth value of $c \in S$. Once the operator is fixed, val_{set} is correlated with the verdict through the K3 rules. For example, $T \vee X = T$, so a True Set value under OR always gives a True verdict. A probe that sees the Set value can use this correlation without the

Table 9. Per-operator stratified probe on the NU subset: the Set-boundary verdict probe within each operator’s samples, compared with the per-operator NU majority. Mean $\pm$ std over 5 seeds. No operator shows clear verdict leakage. NOT is not included because it is unary and its verdict does not use the Set value.

Operator	Probe acc (%)	Per-op NU majority (%)	Δ (pp)
AND	72.08 $\pm$ 0.09	79.16	−7.08
OR	72.68 $\pm$ 0.22	78.89	−6.21
IMP	71.08 $\pm$ 0.30	78.38	−7.29
IFF	51.32 $\pm$ 0.70	50.19	+1.13

Table 10. Operator-early ablation (seed 42): linear verdict and HU decodability (%) at each boundary for the baseline and the operator-early variant, on the same probe set.

Boundary	Verdict		Has-Unknown	
	baseline	op-early	baseline	op-early
Arithmetic	61.0	68.2	80.2	80.2
Order	67.2	67.1	90.8	90.8
Set	69.7	69.2	90.9	90.9
Logic	99.9	99.6	98.0	97.9

boundary encoding the operator or the verdict directly. A probe that uses only the one-hot operator reaches 48.40%, and a probe on the Set-boundary vector plus the operator reaches $61.93 \pm 0.16\%$, only +1.80pp above the Set vector alone. The signals that persist upstream are HU and each engine’s own variable.

6.6 Is the pattern only a result of operator placement?

In THEIA the operator enters only at the Logic Engine, and one may ask whether low upstream verdict decodability comes from this placement alone. We test an *operator-early variant* that adds a learned operator embedding to $\mathbf{c}$, so the operator is visible to the Order, Set and Logic Engines. All else stays the same (2M samples, batch 4096, AdamW lr 10^{-3} , seed 42). The variant reaches 99.56% overall accuracy. Table 10 gives linear probes for both models on the same probe set (50K samples, data seed 999).

With the operator available upstream, verdict decodability rises only at the Arithmetic boundary, where the embedding is added (61.0% $\rightarrow$ 68.2%). At the Order and Set boundaries it changes by at most 0.5pp. All upstream values stay below the 73.4% reference, and the verdict still appears sharply at the Logic boundary. HU does not change upstream. In this single-seed test, moving the operator upstream does not remove the asymmetry. The network still forms the verdict where the two truth values are combined.

6.7 Implementation check with activation patching

Eqs. 1–4 say that the Logic Engine sees only $\mathbf{v}_{\text{ord}}$, $\mathbf{v}_{\text{set}}$ and the operator. We check that the code follows this data flow with activation patching. We build matched $(T \vee F, U \vee F)$ pairs that are identical except for the value that makes the Order result T or U . We put

Table 11. Layer-wise linear probing of the 8-layer post-LN Transformer (matched optimizer settings). Mean $\pm$ std over 5 seeds, with 50K samples per seed. F–T distances are Euclidean.

Layer	SVM Accuracy	F–T Distance
input embed.	0.750 ± 0.001	0.381 ± 0.011
layer 1	0.781 ± 0.007	0.224 ± 0.025
layer 2	0.785 ± 0.011	0.342 ± 0.097
layer 3	0.859 ± 0.040	1.305 ± 0.514
layer 4	0.999 ± 0.000	8.322 ± 3.113
layer 5	1.000 ± 0.000	13.137 ± 0.479
layer 6	1.000 ± 0.000	14.470 ± 0.767
layer 7	1.000 ± 0.000	14.841 ± 0.694
layer 8	1.000 ± 0.000	16.036 ± 0.928
Layer 8 / input ratio	$42.1 \pm 2.5\times$	

the $\mathbf{v}_{\text{ord}}$ of the U-side input into the T-side forward pass. A pair is eligible if the model predicts both unpatched inputs correctly. On one seed, the prediction flips from T to U on 986/986 eligible pairs. Across 5 seeds there are 4,898/4,898 flips. For AND on matched ($T \wedge T$, $U \wedge T$) pairs there are 4,719/4,719 flips. The two inputs of each pair also give a bitwise identical $\mathbf{v}_{\text{set}}$. Because the two inputs share everything except $\mathbf{v}_{\text{ord}}$, a correct implementation must flip every eligible pair. This test therefore confirms the implementation. It is not a separate finding about what the network learned, and testing learned mediation would need pairs that differ in c or S .

6.8 Comparison with the Transformer

We apply the same linear probing protocol to each layer of the 8-layer post-LN Transformer (Table 11). This model reaches the same final correctness as THEIA, but by a different path. Already at its input embeddings, its verdict probe reaches 0.750, above the 73.4% oracle reference. The F–T centroid distance first shrinks and then grows over the layers. The verdict becomes linearly decodable at layer 4. The Transformer’s layers have no fixed domain meaning, so we cannot compute a per-layer ceiling as we did for THEIA’s boundaries. This is the practical difference for explanation. In THEIA, each boundary has a known input set, and we can check the representation against it. In the Transformer, we can only describe the trajectory.

7 Composition over Long Chains (RQ3)

7.1 The mod-3 chain task

Long K3 chains are a weak test of generalization, because K3 has absorbing elements. One F in a long AND chain fixes the result, and later steps cannot change it. Our graph experiment shows this directly. A GNN extension of THEIA reaches 99.99% on transitive order chains from 5 to 50 hops, but 99.6% of these chains are decided by the first edge whose relation is not identity, and the mean position of this edge (counted from 0) is

about 0.5. To test whether accuracy holds when every step matters, we use the *mod-3 chain task*:

$$\text{state}_t = (\text{state}_{t-1} + \text{local_verdict}_t) \bmod 3.$$

This task has no absorbing states and a 9-entry transition table, and its states stay roughly uniform at all depths. The largest deviation from uniform is 0.62pp, at step 100.

Pipeline. At each chain step, a *step module* computes the local verdict of a four-domain question. THEIA’s step module has 1,508,096 parameters, fewer than the 2,751,232 of the four-domain model, because it uses a simpler head and one Logic subspace. A separate transition network with 4,803 parameters maps the previous state and the local verdict to the next state. Gumbel-softmax straight-through discretization (Jang et al. 2017; Maddison et al. 2017; Bengio et al. 2013) rounds the state to a one-hot vector at every step in the forward pass and uses a smooth gradient in the backward pass. As a result, step 500 gets its input in the same form as step 5. Training has three phases. Phase 1 trains the step module alone on 2M single-step samples to $\geq 99.9\%$ *local accuracy* (single-step accuracy), with a plateau-restart shared by all architectures. Phase 2 trains the transition network to 100% (9/9 entries) with teacher forcing. Phase 3 fine-tunes everything end to end on 5-step chains, with a Gumbel temperature that decreases linearly. Of the 5 THEIA seeds, 4 converge in Phase 1 on the first try, and seed 999 needs one plateau-restart (about $2\times$ the Phase-1 compute).

7.2 Backbone comparison

Table 12 changes only the step architecture. Everything else is identical: input encoders, transition network, discretization, three-phase training, data and class weights. The comparators are a flat MLP with 0.80M parameters, a flat MLP with 2.75M parameters, a ResMLP, and the pre-LN Transformer with 3,582,147 parameters. They have $0.53\times$ to $2.4\times$ the parameters of THEIA’s step module, and the extra capacity is on the side of the comparators.

Trained on 5-step chains, THEIA stays at 99.97% or higher at 5–100 steps and reaches $99.96\% \pm 0.04\%$ at 500 steps. Its per-seed values at 500 steps are {99.90, 99.99, 99.97, 99.99, 99.96}%. The strict aggregate without seed 999 is also $99.96\% \pm 0.04\%$. The flat MLPs match THEIA’s Phase 1 accuracy within 0.04%, but they collapse to chance ($\sim 33\%$). Going from 0.80M to 2.75M parameters does not change this.

ResMLP grid. We test a 2×2 grid of ResMLPs with 4 or 8 blocks and a hidden-width expansion of 2 or 4 (written 2d or 4d), 5 seeds each (Table 13). Each block is Linear $\rightarrow$ GELU $\rightarrow$ LayerNorm $\rightarrow$ Linear with a skip connection. Three configurations have about 2.78M parameters, and 4-block $\times$ 4d has 2.85M. The ResMLP partly recovers, and depth has a strong effect. The 4-block configurations reach means of 86.15–87.49%, and the 8-block configurations reach 97.95–97.97%. At depth 8, the expansion has almost no effect. But over the whole grid, only 3/20 trials reach $\geq 99\%$ (2/19 without the one seed that needed a plateau-restart). No single configuration reaches $\geq 99\%$ on a majority of seeds. A one-sided Fisher exact test of the rate of seeds at $\geq 99\%$, THEIA (5/5) against

Table 12. Backbone comparison under the same three-phase Gumbel-softmax training (5 seeds per architecture unless noted, accuracy in %). The Phase 1 row is accuracy on the single-step training samples. The ResMLP column is the 4-block $\times$ 4d configuration; its 500-step value is the strict 4-seed aggregate without the one seed that needed a plateau-restart (as-specified 5-seed value: 86.15 ± 9.17). THEIA’s strict 4-seed aggregate equals its 5-seed value. The best grid configuration, 8-block $\times$ 2d, reaches 97.97 ± 1.43 at 500 steps (1/5 seeds $\geq 99\%$).

	THEIA (1.51M)	MLP (0.80M)	MLP (2.75M)	ResMLP (2.85M)	Pre-LN TF (3.58M)
Phase 1	99.94	99.91	99.90	99.91	99.92
Phase 2	100	100	100	100	100
5-step	99.97	80.44	76.88	99.76	99.99
10-step	99.97	66.92	61.85	99.58	99.98
50-step	99.97	35.13	34.36	98.00	99.94
100-step	99.97	33.32	33.54	96.05	99.86
500-step	99.96 $\pm$ 0.04	33.58	33.35	82.83 ± 6.22	99.24 $\pm$ 0.34

Table 13. ResMLP 2 $\times$ 2 depth $\times$ expansion grid at 500 steps. The hidden dimension d is chosen for a near-matched parameter count. Seeds are the same as for THEIA. $\geq 99\%$: seeds that reach the reliability threshold. $\dagger$: seed 999 of 4b $\times$ 4d needed a plateau-restart and reached 99.41%; without it the strict aggregate is 82.83 ± 6.22 with 0/4 $\geq 99\%$.

Config	Params	500-step	Range	$\geq 99\%$
4b $\times$ 4d ($d=280$)	2,848,547	$86.15 \pm 9.17^\dagger$	75.27–99.41	1/5 †
4b $\times$ 2d ($d=383$)	2,776,047	87.49 ± 19.16	53.27–97.71	0/5
8b $\times$ 2d ($d=276$)	2,774,659	97.97 ± 1.43	95.55–99.27	1/5
8b $\times$ 4d ($d=198$)	2,780,707	97.95 ± 2.03	94.34–99.12	1/5
Aggregate: 3/20 trials $\geq 99\%$ (as-specified), 2/19 (strict)				

the pooled grid (3/20), gives $p \approx 1.1 \times 10^{-3}$. With THEIA’s strict count (4/4) against 2/19, $p \approx 1.7 \times 10^{-3}$.

Pre-LN Transformer. The pre-LN Transformer reaches $99.24\% \pm 0.34\%$ at 500 steps, with 3/5 seeds $\geq 99\%$ and a per-seed range of 98.85–99.66%. Attention can also give length generalization under discretized end-to-end training. At the $\geq 99\%$ threshold of H1, THEIA is the only tested architecture that passes on every seed. With 5 seeds, the difference from the Transformer (5/5 vs. 3/5) is not significant (one-sided Fisher exact test, $p = 0.22$). At $\geq 99.9\%$, only THEIA passes on any seed (5/5 vs. 0/5, $p = 0.004$). This threshold was not part of H1, so this second comparison is exploratory. For a user who trains a single model, the lowest seed matters most. THEIA’s lowest seed is 99.90%, the Transformer’s is 98.85%, and the best ResMLP configuration’s is 95.55%. THEIA’s cross-seed std is about $9\times$ smaller than the Transformer’s (0.04% vs. 0.34%).

We call this ordering of step architectures, under our pipeline hyperparameters, the *reliability spectrum*: flat MLPs fail, ResMLPs partly recover, the pre-LN Transformer is near-perfect on most seeds, and THEIA is near-perfect on every seed.

Table 14. Discretization operator and end-to-end training ablation on the mod-3 chain. Mean $\pm$ std over 3 seeds {42, 123, 256} (population std). In the 500-step column, “w” is the worst single-seed value. Post-P3 local is the step module’s single-step accuracy on new samples after Phase 3 (after Phase 2 when Phase 3 is skipped). The frozen-step and no-Phase-3 rows are identical, because their step and transition networks make the same argmax decisions on the evaluation chain.

Mode	Backbone	5-step (%)	500-step (%)	post-P3 local (%)
<i>(a) Discretization operator, THEIA step module</i>				
Gumbel-ST (default)	THEIA	100.00 $\pm$ 0.00	99.98 $\pm$ 0.02 (w99.95)	100.00 $\pm$ 0.00
Soft (no discretization)	THEIA	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00 (w99.99)	100.00 $\pm$ 0.00
Hard argmax (no ST grad)	THEIA	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00 (w100.00)	100.00 $\pm$ 0.00
Fixed temp $\tau=1.0$	THEIA	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00 (w100.00)	100.00 $\pm$ 0.00
Fixed temp $\tau=0.1$	THEIA	100.00 $\pm$ 0.00	99.98 $\pm$ 0.01 (w99.97)	100.00 $\pm$ 0.00
<i>(b) Discretization operator, flat-MLP step module (2.75M)</i>				
Gumbel-ST (default)	Flat MLP	76.24 $\pm$ 0.59	33.85 $\pm$ 0.24 (w33.57)	94.24 $\pm$ 0.09
Soft (no discretization)	Flat MLP	77.50 $\pm$ 0.59	33.40 $\pm$ 0.34 (w33.03)	94.56 $\pm$ 0.15
Hard argmax (no ST grad)	Flat MLP	73.52 $\pm$ 0.18	33.42 $\pm$ 0.55 (w32.75)	93.45 $\pm$ 0.09
<i>(c) End-to-end Phase 3 removed, THEIA step module</i>				
Frozen step (P3 = transition only)	THEIA	99.66 $\pm$ 0.25	75.05 $\pm$ 17.09 (w52.59)	99.93 $\pm$ 0.06
No Phase 3	THEIA	99.66 $\pm$ 0.25	75.05 $\pm$ 17.09 (w52.59)	99.93 $\pm$ 0.06
<i>(d) End-to-end Phase 3 removed, flat-MLP step module</i>				
Frozen step (P3 = transition only)	Flat MLP	72.28 $\pm$ 0.20	33.25 $\pm$ 0.28 (w32.93)	93.02 $\pm$ 0.06
No Phase 3	Flat MLP	72.28 $\pm$ 0.20	33.25 $\pm$ 0.28 (w32.93)	93.02 $\pm$ 0.06

7.3 What drives the 500-step result

One could expect the discretization to be the cause. It rounds every state to one-hot, so it might stop errors from piling up for any step module. We test this with an ablation over the discretization operator and over end-to-end training (Table 14, 3 seeds). As a heuristic *compounding reference*, if each step were correct independently with probability 0.999, all 500 steps would be correct with probability $0.999^{500} \approx 60.6\%$. This is not a strict bound, because two errors can cancel in a mod-3 sum.

The discretization operator is not necessary. In *soft state propagation*, the state is passed as a probability vector and is never rounded, and THEIA still reaches 100.00% at 500 steps. The hard-argmax variant and both fixed temperatures also reach $\geq 99.98\%$. The operator is also not sufficient. The flat MLP collapses to chance ($\sim 33\%$) by 500 steps under every operator, including the default straight-through operator.

Two factors explain the result. The first is a step architecture that keeps $\sim 100\%$ local accuracy. The flat MLP reaches only 93.02% on new single-step samples without Phase 3 and $\sim 94\%$ after it. With independent errors at this rate, all 5 steps would be correct with probability $0.9424^5 \approx 74\%$, close to its measured 5-step accuracy of 76.24% in the same runs. The flat MLP fails because of low local accuracy, and the failure is not specific to chain length. The second factor is end-to-end Phase 3 training. If we freeze the step module or skip Phase 3, THEIA’s local accuracy is 99.93% and its 500-step accuracy drops to 75.05%, with a worst seed of 52.59%. This is close to the compounding reference at that local accuracy ($0.9993^{500} \approx 70.5\%$). With Phase 3, THEIA’s local

Table 15. Multi-hop SQL-NULL propagation: verdict accuracy vs. depth L (mean $\pm$ std over 3 seeds), and the unsafe-verdict rate ($U \rightarrow \text{definite}$).

Depth L	THEIA	Transformer	Flat MLP
5	100.00 $\pm$ 0.00	99.99 $\pm$ 0.00	91.13 $\pm$ 1.11
20	100.00 $\pm$ 0.00	99.98 $\pm$ 0.01	90.48 $\pm$ 1.52
50	100.00 $\pm$ 0.00	99.98 $\pm$ 0.01	90.60 $\pm$ 1.23
100	100.00 $\pm$ 0.00	99.99 $\pm$ 0.01	90.52 $\pm$ 1.35
200	100.00 $\pm$ 0.00	99.99 $\pm$ 0.01	90.56 $\pm$ 1.24
500	100.00 $\pm$ 0.00	99.97 $\pm$ 0.01	90.60 $\pm$ 1.15
<i>Unsafe $U \rightarrow \text{definite}$ (%), mean over 3 seeds</i>			
5	0.00	0.00	4.34
20	0.00	0.01	5.92
50	0.00	0.02	5.70
100	0.00	0.00	5.70
200	0.00	0.01	5.86
500	0.00	0.03	5.53

accuracy is 100.00%. End-to-end training thus pushes the per-step error close to zero, and this is what the 500-step result needs.

Locality. The mod-3 chain task is local: each step can be computed from local inputs. Modular arithmetic on graphs (mod 5) instead needs global aggregation over the whole input, and our GNN extension could not learn it. In our experiments, what decides success is this *computation locality*, and not chain length alone.

7.4 External check: multi-hop SQL-NULL propagation

SQL handles NULL with three-valued logic, which is K3. NULL matters most in derived-column pipelines, where each column is computed from earlier ones, so NULL propagates along the chain. We build a *multi-hop SQL-NULL test*. Each row is one pipeline with L hops. Hop t computes a local K3 verdict ℓ_t from its own operands with the four-domain step, with NULL inputs at rate $p_{\text{null}} = 0.05$. The state starts at $s_0 = \ell_0$ and is updated as $s_t = \text{FOLD}(\text{op}_t, s_{t-1}, \ell_t)$, where FOLD applies a per-hop operator $\text{op}_t \in \{\text{AND}, \text{OR}, \text{XOR}, \text{XNOR}\}$ under strong-Kleene semantics. XOR and XNOR are not monotone, so no simple global reduction solves the task. We use $L \in \{5, 20, 50, 100, 200, 500\}$, 3 seeds, and 10,000 held-out rows per depth per seed. The step modules are THEIA, the pre-LN Transformer and a flat MLP with a comparable parameter count. The last two are *monolithic*, that is, one network with no separate engines. An *unsafe verdict* is a T or F output when the correct final verdict is U.

THEIA keeps 100.00% verdict accuracy at every depth up to 500, with 0.00% unsafe verdicts (Table 15). The pre-LN Transformer is almost the same, with 99.97–99.99% accuracy and at most 0.03% unsafe verdicts, so this test does not separate THEIA from the Transformer. The flat MLP has about 95% local accuracy and a persistent deficit: about 90.6% accuracy and 4.3–5.9% unsafe verdicts at every depth. In practice, it gives a definite answer for about 1 in 20 columns whose correct value is NULL. It does not

collapse to chance here, because a K3 fold can recover from errors. An absorbing event (AND with False, OR with True) resets the state to the correct value. For this reason the task is a weaker test of depth than the mod-3 chain, and its value is in the NULL-safety measure.

8 Discussion

8.1 *What THEIA gives that a Transformer does not*

Both THEIA and a tuned Transformer learn the full K3 table. The difference is in what we can see and check. THEIA’s engines have a fixed data flow, so each boundary has a known set of inputs. This lets us compute the Bayes ceiling of the Unknown signal at each boundary from the flag structure, and we find that the network reaches it within 0.3pp (Table 7). It lets us predict that the operator is not decodable upstream, and the probes confirm it (Table 5). It lets us ask whether the verdict leaks upstream in a nonlinear or per-operator form, and the controls say no (Tables 6, 8 and 9). These are explanations that are checked against symbolic structure, not only against output labels. The Transformer reaches the same answers, but its verdict is already partly decodable at its input embeddings, and its layers have no fixed meaning that we can compute a ceiling for (Table 11).

The second difference is reliability. On the discretized mod-3 chain, the lowest THEIA seed is at 99.90% after 500 steps. The pre-LN Transformer has a higher spread, and 2 of 5 seeds fall below 99%. For a system that must be trusted over many steps, the worst seed is the relevant number.

8.2 *Relevance to explainable neurosymbolic AI*

THEIA is a neural model with a symbolic-looking structure. Its engines correspond to the parts of a logical expression, and its prototypes correspond to the three truth values. But the K3 rules are not written into it. They are learned. This combination makes two kinds of explanation possible. First, we can read intermediate truth values from the boundaries (Order value at Order, Set value at Set), so a verdict can be traced back to the sub-results that produced it. Second, we can test explanation claims quantitatively, with ceilings that come from the symbolic task and not from the model. We think this is a useful pattern for explainable neurosymbolic models in general: design the architecture so that each module has a known input set, and then check the learned representation against what that input set allows.

8.3 *Task–architecture alignment*

The four benchmark domains map one-to-one onto the four engines of THEIA, and we chose this alignment in the design. For this reason we tested whether the results depend on it. We trained three misaligned variants on the four-domain task (3 seeds each): one engine shared by the Order and Set domains ($m=3$ engines for $n=4$ domains), an extra engine with no domain ($m=5$), and a deranged map where every engine handles the wrong domain (same parameter count). Table 16 shows that misalignment has little

Table 16. Task–architecture misalignment (3 seeds each): number of learned K3 rules (of 39) and overall accuracy.

Variant	Params	K3 rules /39	Overall acc. (%)
aligned ($m=4$)	2.75M	39.0 ± 0.0	99.95 ± 0.01
merged Order/Set ($m=3$)	2.55M	38.7 ± 0.5	99.96 ± 0.01
extra engine ($m=5$)	3.02M	38.0 ± 1.4	99.93 ± 0.02
deranged map ($m=4$)	2.75M	38.3 ± 0.9	99.94 ± 0.02

effect. Overall accuracy changes by less than 0.1pp. The number of learned K3 rules (mean over 3 seeds; a rule counts as learned if its accuracy is $> 99\%$) goes from 39.0/39 to 38.0–38.7/39, with worst-rule accuracy 98.6–99.2%. Four-domain accuracy therefore does not depend on the domain-to-engine map. We did not run these variants on the chain task.

8.4 Limitations

All experiments use synthetic benchmarks, which give exact ground truth and precise control over task complexity (Veličković et al. 2022; Rodionov and Prokhorenkova 2025). The SQL-NUL test is a first step toward real tasks, but full real-database workloads are future work. Numeric range generalization fails sharply at $5\times$ the training range (Trask et al. 2018), so our mechanistic claims concern a bounded-range representation. Global computation, such as modular arithmetic on graphs, is out of reach for our local message passing. The mod-3 chain uses a minimal 3-state space, and richer state spaces (mod- k , automaton traces) are untested. Our ResMLP grid covers 4 configurations at one parameter scale, and we did not test deeper models (≥ 12 blocks) or other normalization placements. The chain-pipeline hyperparameters (Phase-3 learning rate, temperature schedule, output head) were set in THEIA runs and not tuned per baseline, and we did not isolate which THEIA component drives the 500-step result. For training time, neither model was tuned symmetrically. Finally, our probe results hold for linear and MLP probes; much stronger probe families were not tested.

9 Conclusion

THEIA learns the complete K3 truth table from data, including all absorption rules, with no K3 gates and no symbolic solver. A tuned Transformer also learns the table, so our main results are about what the modular design makes visible and reliable. Inside THEIA, the Unknown signal is decodable at every engine boundary, at or near the Bayes ceiling that the data flow allows, while the verdict forms only at the Logic Engine, where the operator enters. On long discretized chains, step architectures form a reliability spectrum, and THEIA is the only one that stays at or above 99.9% on every seed at 500 steps. Ablations show that this comes from the step architecture and end-to-end training, not from discretization. We hope these results support modular, checkable designs for neural reasoning under missing information. Future work includes richer state spaces, first-order

and other non-classical logics, real database benchmarks, and a symmetric tuned-versus-tuned comparison.

Declaration of conflicting interests

The author declares no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author received no financial support for the research, authorship, and/or publication of this article.

Data and code availability

All data are synthetic and are generated by the released code. The code, data generators, diagnostics and analysis scripts are available at <https://github.com/xxx12e/theia-k3>.

Prior versions

A preprint of this work is available on arXiv (arXiv:2604.11284). A short version was presented as a poster at an ICML 2026 workshop without archival proceedings. This article is not under review elsewhere.

Use of AI tools

The author designed all research. AI tools were used to write code and to edit the English of the manuscript.

Supplementary material

Supplementary material is available with this article. It contains the full hypotheses and analysis choices (A), diagnostic construction and per-rule frequencies (B), architecture and training details (C), additional probing results (D), activation-patching details (E), chain-pipeline details (F), tuned-Transformer details (G), and reproducibility notes (H).

References

- Alain G and Bengio Y (2017) Understanding intermediate layers using linear classifier probes. In: *ICLR Workshop*.
- Andreas J, Rohrbach M, Darrell T and Klein D (2016) Neural module networks. In: *CVPR*.
- Anil C, Wu Y, Andreassen A, Lewkowycz A, Misra V, Ramasesh V, Slone A, Gur-Ari G, Dyer E and Neyshabur B (2022) Exploring length generalization in large language models. In: *NeurIPS*.

- Badreddine S, d'Avila Garcez AS, Serafini L and Spranger M (2022) Logic tensor networks. *Artificial Intelligence* 303: 103649.
- Bengio Y, Léonard N and Courville A (2013) Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432.
- Chan SC, Hsu LS, Brody S and Teh HH (1989) Neural three-valued-logic networks. In: *IJCNN*.
- Chen D, Lin Y, Li W, Li P, Zhou J and Sun X (2020) Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: *AAAI*.
- Csordás R, Irie K and Schmidhuber J (2022) The neural data router: Adaptive control flow in Transformers improves systematic generalization. In: *ICLR*.
- Damera SS, Matheu R, Puranic AG and Baras JS (2026) Polynomial surrogate training for differentiable ternary logic gate networks. arXiv:2603.00302.
- de Moura L and Bjørner N (2008) Z3: An efficient SMT solver. In: *TACAS*.
- Delétang G, Ruoss A, Grau-Moya J, Genewein T, Wenliang LK, Catt E, Cundy C, Hutter M, Legg S, Veness J and Ortega PA (2023) Neural networks and the Chomsky hierarchy. In: *ICLR*.
- Dong H, Mao J, Lin T, Wang C, Li L and Zhou D (2019) Neural logic machines. In: *ICLR*.
- Dziri N, Lu X, Sclar M, Li XL, Jiang L, Lin BY, West P, Bhagavatula C, Le Bras R, Hwang JD, Sanyal S, Welleck S, Ren X, Ettinger A, Harchaoui Z and Choi Y (2023) Faith and fate: Limits of Transformers on compositionality. In: *NeurIPS*.
- Elhage N, Nanda N, Olsson C, Henighan T, Joseph N, Mann B, Askell A, Bai Y, Chen A, Conerly T, DasSarma N, Drain D, Ganguli D, Hatfield-Dodds Z, Hernandez D, Jones A, Kernion J, Lovitt L, Ndousse K, Amodei D, Brown T, Clark J, Kaplan J, McCandlish S and Olah C (2021) A mathematical framework for Transformer circuits. *Transformer Circuits Thread*.
- Fitting M (1985) A Kripke–Kleene semantics for logic programs. *Journal of Logic Programming* 2(4): 295–312.
- Geiger A, Lu H, Icard T and Potts C (2021) Causal abstractions of neural networks. In: *NeurIPS*.
- Gilmer J, Schoenholz SS, Riley PF, Vinyals O and Dahl GE (2017) Neural message passing for quantum chemistry. In: *ICML*.
- Hsu LS, Loe KF, Chan SC and Teh HH (1991) Two-valued neural logic network. *Proceedings of SPIE* 1469: 197–207.
- Hsu LS, Teh HH, Chan SC and Loe KF (1990) Multi-valued neural logic networks. In: *ISMVL*. pp. 426–432.
- Jang E, Gu S and Poole B (2017) Categorical reparameterization with Gumbel-softmax. In: *ICLR*.
- Kleene SC (1952) *Introduction to Metamathematics*. North-Holland.
- Lake BM and Baroni M (2018) Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In: *ICML*.
- Li Q, Han Z and Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: *AAAI*.
- Li Z, Huang J and Naik M (2023) Scallop: A language for neurosymbolic programming. In: *PLDI*.
- Liu B, Ash JT, Goel S, Krishnamurthy A and Zhang C (2023) Transformers learn shortcuts to automata. In: *ICLR*.
- Loshchilov I and Hutter F (2019) Decoupled weight decay regularization. In: *ICLR*.
- Loukas A (2020) What graph neural networks cannot learn: Depth vs. width. In: *ICLR*.

- Maddison CJ, Mnih A and Teh YW (2017) The concrete distribution: A continuous relaxation of discrete random variables. In: *ICLR*.
- Manhaeve R, Dumančić S, Kimmig A, Demeester T and De Raedt L (2021) Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence* 298: 103504.
- Marra G, Dumančić S, Manhaeve R and De Raedt L (2024) From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence* 328: 104062.
- Meng K, Bau D, Andonian A and Belinkov Y (2022) Locating and editing factual associations in GPT. In: *NeurIPS*.
- Nanda N, Chan L, Lieberum T, Smith J and Steinhardt J (2023) Progress measures for grokking via mechanistic interpretability. In: *ICLR*.
- Petersen F, Borgelt C, Kuehne H and Deussen O (2022) Deep differentiable logic gate networks. In: *NeurIPS*.
- Petersen F, Kuehne H, Borgelt C, Welzel J and Ermon S (2024) Convolutional differentiable logic gate networks. In: *NeurIPS*.
- Press O, Smith NA and Lewis M (2022) Train short, test long: Attention with linear biases enables input length extrapolation. In: *ICLR*.
- Riegel R, Gray A, Luus F, Khan N, Makondo N, Akhalwaya IY, Qian H, Fagin R, Barahona F, Sharma U, Ikbal S, Karanam H, Neelam S, Likhyan A and Srivastava S (2020) Logical neural networks. arXiv:2006.13155.
- Rodionov G and Prokhorenkova L (2025) Discrete neural algorithmic reasoning. In: *ICML*.
- Serafini L and d'Avila Garcez AS (2016) Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In: *NeSy Workshop*.
- Snell J, Swersky K and Zemel R (2017) Prototypical networks for few-shot learning. In: *NeurIPS*.
- Teh HH (1995) *Neural Logic Networks*. Singapore: World Scientific.
- Trask A, Hill F, Reed S, Rae J, Dyer C and Blunsom P (2018) Neural arithmetic logic units. In: *NeurIPS*.
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L and Polosukhin I (2017) Attention is all you need. In: *NeurIPS*.
- Veličković P, Puigdomènech Badia A, Budden D, Pascanu R, Banino A, Dashevskiy M, Hadsell R and Blundell C (2022) The CLRS algorithmic reasoning benchmark. In: *ICML*.
- Vig J, Gehrmann S, Belinkov Y, Qian S, Nevo D, Singer Y and Shieber S (2020) Investigating gender bias in language models using causal mediation analysis. In: *NeurIPS*.
- Wang G and Shi H (1998) TMLNN: Triple-valued or multiple-valued logic neural network. *IEEE Transactions on Neural Networks* 9(6): 1099–1117.
- Xiong R, Yang Y, He D, Zheng K, Zheng S, Xing C, Zhang H, Lan Y, Wang L and Liu TY (2020) On layer normalization in the Transformer architecture. In: *ICML*.
- Xu J, Zhang Z, Friedman T, Liang Y and Van den Broeck G (2018) A semantic loss function for deep learning with symbolic knowledge. In: *ICML*.
- Xu K, Li J, Zhang M, Du SS, Kawarabayashi Ki and Jegelka S (2020) What can neural networks reason about? In: *ICLR*.
- Yang Z, Ishay A and Lee J (2020) NeurASP: Embracing neural networks into answer set programming. In: *IJCAI*.