From Grokking to Micro-Routing: Graph Neural Cellular Automata for Sub-Millisecond Agent Tool Selection

Julian M. Kleber
Independent Researcher
julian.m.kleber@gmail.com

October 2026

Abstract

In autonomous software agents, deciding which tool to invoke—such as reading a file, searching a database, or replying to the user—is fundamentally a micro-routing decision. Today's systems dispatch every step through multi-billion-parameter Large Language Models (LLMs), spending seconds of latency and thousands of redundant tokens merely to select a function from a list. In this paper, I demonstrate that Neural Cellular Automata (NCAs) offer an ultra-lightweight alternative: a 59.2k-parameter network (35.3k in compact K=4 / SVD64 configurations) that functions as a fast reflex arc. Instead of generating tokens sequentially, my **Graph Neural Cellular Automata (GNCA)** framework parameterizes local cellular communication via continuous **architecture meta-learning**. Nodes representing user goals, execution history, and argument slots exchange local state updates over fast recurrence cycles to reach rapid consensus. I establish structural invariants governing graph subsumption, receptive field diameter ($O(1)$ vs $\Omega(V)$), and permutation symmetry, formally verifying all proofs in **Lean 4** with zero unproven axioms to certify the cellular search space. Continuous architecture meta-learning discovers relational topologies reaching 65.33% top-1 (35.3k parameters), outperforming 1D regular lattices (64.64%) and coming within 0.54 pp of a full-attention Transformer baseline (65.87%, 1.2M parameters). Scaling feature capacity to SVD128 with $K = 8$ recurrence and label smoothing under extended training (59.2k parameters) induces delayed generalization (grokking), reaching **68.07**% test top-1 accuracy (95% CI: $[66.49, 69.60]$%), closely approaching competitive tuned tabular ensembles within 1.4 pp (Random Forest 67.88%, HistGradientBoosting 69.47%) while running in sub-millisecond CPU latency (0.66 ms). Deployed as a System-1 micro-router paired with an LLM actuator in a Dual-System architecture, GNCA achieves an average routing latency of **21.43** ms across 89 live benchmark turns on a local CPU. For reference, cloud-hosted LLM APIs took 950–4,409 ms per routing call end to end; this gap includes network round-trips and is a deployment comparison, not a same-hardware model comparison. In exploratory pilot diagnostic evaluations ($N = 10$ tasks on Sierra τ^2-bench and $N = 30$ probes on BFCL v4), the hybrid agent reaches an **80.0**% Pass[1] success rate (8/10 tasks) and **90.0**% BFCL v4 irrelevance triage (27/30 probes, 95% Wilson CI: $[74.4\%, 96.5\%]$), while reducing measured cumulative token consumption by **44.0**% (209,734 → 117,462 tokens; 92,272 saved).

1 Introduction

Autonomous software agents operate as iterative closed-loop systems: observing environment feedback (compiler diagnostics, file diffs, test logs) and synthesizing next actions. In biological nervous systems, routine motor actions do not require deliberative processing in the cerebral cortex; a low-level reflex arc in the spinal cord coordinates rapid responses in milliseconds. In contrast, modern AI agents dispatch every mundane operation through massive, multi-billion-parameter autoregressive models. This design causes critical bottlenecks:

1. **Latency and Memory Pressure.** Generating tokens sequentially causes high latency and heavy memory traffic, slowing down agent loops.
2. **Context Window Bloat.** Intermediate steps fill context windows with repetitive JSON schema and syntax clutter.
3. **Rigid 1D Assumptions.** Prior cellular models restrict computation to 1D regular lattices, forcing signals to crawl one step at a time like a game of telephone, ignoring that agent execution traces form multi-scale hierarchical trees.

Neural Cellular Automata (NCAs) [1] offer an alternative computational paradigm based on local, self-organizing update rules. Historically, NCAs were studied in continuous 2D grids for biological morphogenesis, texture generation, and self-repair. However, their core inductive biases—iterative spatial message passing across discrete cells governed by uniform update rules—align naturally with structured sequential decision processes. Like biological cells communicating with their immediate neighbors to form complex tissues, NCA cells update

their local states in parallel without global bottlenecking. Recent investigations (e.g., Mittal et al. [5]) probed token-level text NCAs, but their application to structured multi-action tool execution has remained limited to rigid sequential chains.

Contributions. In this paper, I extend Neural Cellular Automata beyond 1D topologies into a meta-learned graph framework for hierarchical agent routing. My core contributions are fourfold:

- **Graph Cellular Automata:** I formulate GNCA, replacing fixed 1D convolutional stencils with continuous architecture search over graph topologies, reaching 65.33% top-1 at $K = 4$ and 68.07% via grokking with deep recurrence.
- **Machine-Checked Structural Invariants:** I formally verify in Lean 4 that 1D lattices embed into GNCA, prove an exponential diameter gap ($O(1)$ vs $\Omega(V)$), and prove permutation equivariance with zero unproven axioms.
- **Dual-System Architecture:** I introduce a high-throughput System-1/System-2 agent harness with risk-gated dynamic schema pruning, filtering non-candidate tools before LLM actuation.
- **Pilot Agentic Diagnostics:** In exploratory proof-of-concept deployments ($N = 10$ tasks on τ^2-bench, $N = 30$ BFCL v4 irrelevance probes, and one SWE episode), GNCA achieves 80.0% Pass[1] (8/10), 90.0% irrelevance triage (27/30), and routes in 21.43 ms on an Apple M1 CPU (cloud LLM API calls, measured end to end including network: 950–4,409 ms; not hardware-matched).

2 Related Work

Neural Cellular Automata and Reasoning. Mordvintsev et al. [1] introduced Growing Neural Cellular Automata, demonstrating self-organizing pattern morphogenesis and damage recovery. Subsequent work extended NCAs to multi-scale hierarchical structures [2] and decentralized algorithmic reasoning [4]. Grattarola et al. [3] introduced Learning Graph Cellular Automata (LGCA), showing that GNNs can execute cellular update rules over fixed, predefined graphs (e.g., Voronoi meshes). In language representation, Mittal et al. [5] and Lee et al. [6] demonstrated that cellular rules can learn 1D token representations. My framework departs fundamentally from token generation and static GCA: agent execution traces lack a predefined graph, requiring bilevel architecture meta-learning to discover optimal communication graphs, while introducing cross-turn state carry and formal Lean 4 verification for hierarchical tool routing.

Tool Calling, Agent Harnesses, and Pre-Routing. Tool use in autonomous systems is evaluated on benchmarks like the Berkeley Function Calling Leaderboard (BFCL) [7] and SWE-bench. Production agent harnesses execute many typed System-1 decisions per task [10]. Singla et al. [9] showed that pre-inference action routing cuts prompt latency and token overhead by half. Expanding on these findings, real agents must handle three coupled tasks: detecting irrelevance to abstain gracefully, binding argument slots to workspace entities, and coordinating multi-turn plans.

Architecture Meta-Learning & Graph Bottlenecks. Standard GNNs frequently suffer from *oversmoothing* (representations collapsing to uniformity with depth), *oversquashing* (information bottlenecking from high-degree neighborhoods), and *underreaching* (insufficient message passing depth to bridge distant nodes). By casting topology optimization as bilevel architecture meta-learning, my framework discovers sparse relational graphs that balance local message passing with long-range goal broadcasting.

3 Task Formulation & Architecture

3.1 Hierarchical Action Formulation

Rather than treating tool selection as a flat classification problem, I formulate agent dispatch as synthesizing an action tuple.

$$\mathbf{a}_T = (y_{\text{tool}},\ \mathbf{y}_{\text{arg}},\ g_{\text{abstain}},\ \tau_{\text{escalate}}) \tag{1}$$

The four tuple components address the complete agent action interface.

- $y_{\text{tool}} \in \mathcal{V}$: Discrete tool identifier chosen from vocabulary $|\mathcal{V}| = 13$ (e.g., `read_file`, `edit_file`, `run_command`, `list_dir`, and terminal completion `respond`).
- $\mathbf{y}_{\text{arg}} \in \mathcal{A}$: Structured argument slot predictions mapping discrete entity slots in the active workspace context.
- $g_{\text{abstain}} \in [0, 1]$: Calibrated scalar logit indicating whether the agent should abstain from calling external tools.
- $\tau_{\text{escalate}} \in [0, 1]$: Predictive execution risk score for routing challenging turns to human verifiers.

3.2 Beyond 1D: Graph Neural Cellular Automata

In standard 1D NCAs, cell interactions are constrained to nearest neighbors $\{i-1, i, i+1\}$ along a rigid sequence grid. In agent workflows, however, execution traces exhibit multi-scale relational dependencies: the root goal directly conditions argument extraction, while recursive sub-agent tools branch hierarchically.

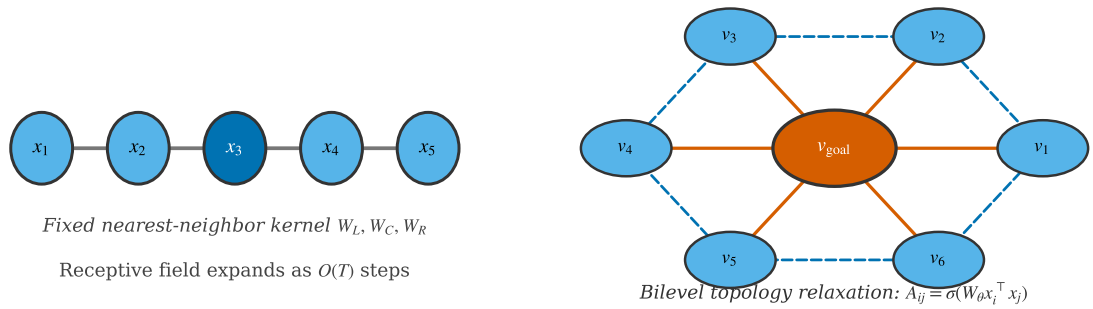


Figure 1: **Architectural schematic and topological comparison across cellular interaction structures.** (a) Traditional 1D cellular automata restrict message passing strictly to adjacent linear neighbors, requiring $\Omega(V)$ sequential updates for global goal integration across execution traces. (b) The meta-learned Graph NCA framework relaxes topology via continuous bilevel optimization, parameterizing edge weights as $A_{ij} = \sigma(W_\theta x_i^\top x_j)$. This enables direct goal broadcasting with graph diameter ≤ 2 ($O(1)$ receptive field growth) and permutation-equivariant state updates across agent execution nodes.

I generalize the cellular substrate to an arbitrary relational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $|\mathcal{V}| = V = 5$ nodes represent the causal trace history window ($L = 5$, with node 0 as the root goal anchor and nodes $1 \dots 4$ as chronological execution turns). In concrete terms, each node maintains a compact 16-number scratchpad ($h_i^{(t)} \in \mathbb{R}^{16}$, initialized to $\mathbf{0}$). In each cycle, nodes whisper state updates to neighboring cells across $K = 4$ fast iterations, reaching collective consensus without global matrix bottlenecking.

Continuous Spatial Perception. Let $A \in \mathbb{R}^{V \times V}$ denote the learned adjacency and $\tilde{A} = \tilde{D}^{-1/2}(A + I)\tilde{D}^{-1/2}$ be the symmetrically normalized adjacency with self-loops ($\tilde{D}_{ii} = 1 + \sum_j A_{ij}$). Symmetrized $\tilde{A}$ acts as a spatial low-pass filter (local consensus smoothing), while the normalized Laplacian $\tilde{L} = I - \tilde{A}$ acts as a complementary high-pass divergence operator (detecting state contrast between adjacent cells), directly mirroring the Sobel and Laplacian perception filters in classical NCAs [1]. The spatial perception vector $p_i^{(t)} \in \mathbb{R}^{3H}$ concatenates cell state, smoothed consensus, and difference features:

$$p_i^{(t)} = \left[h_i^{(t)},\ \sum_{j=1}^{V} \tilde{A}_{ij} W_A h_j^{(t)},\ \sum_{j=1}^{V} \tilde{L}_{ij} W_L h_j^{(t)} \right] \tag{2}$$

where $W_A, W_L \in \mathbb{R}^{H \times H}$ are independent linear projections parameterizing low- and high-pass perception channels.

Pointwise State Update. The perceived vector $p_i^{(t)}$ is concatenated with the cell's fixed observation $x_i \in \mathbb{R}^D$, forming $z_i^{(t)} = [p_i^{(t)}; x_i] \in \mathbb{R}^{3H+D}$. A shared pointwise update network computes the state delta $\Delta h_i^{(t)} \in \mathbb{R}^H$:

$$\Delta h_i^{(t)} = W_2\ \mathrm{GELU}(W_1 z_i^{(t)} + b_1) + b_2 \tag{3}$$

where $W_1 \in \mathbb{R}^{D_{\text{mid}} \times (3H+D)}$, $W_2 \in \mathbb{R}^{H \times D_{\text{mid}}}$, and $D_{\text{mid}} = 96$ (with $D = 82$ yielding 35.3k parameters in SVD64, and $D = 146$ yielding 59.2k in SVD128).

Residual State Accumulation. Updates are applied residually across $K = 4$ cycles:

$$h_i^{(t+1)} = h_i^{(t)} + m_i^{(t)} \odot \Delta h_i^{(t)} \tag{4}$$

where $m_i^{(t)} \sim \text{Bernoulli}(p)$ ($p = 0.8$ during training; 1.0 at inference) enforces asynchronous cellular computation.

3.3 Architecture Meta-Learning on Spatial Topologies

To discover the optimal interaction topology without manual heuristic tuning, I formulate a bilevel architecture meta-learning problem.

$$\begin{aligned} &\min_{\alpha \in \mathbb{R}^{V \times V}} \mathcal{L}_{\text{val}}\left(\theta^*(\alpha);\ A^{(\alpha)}\right) \\ &\text{s.t. } \theta^*(\alpha) = \arg\min_{\theta} \mathcal{L}_{\text{train}}(\theta;\ A^{(\alpha)}) \end{aligned} \tag{5}$$

The continuous matrix $\alpha \in \mathbb{R}^{V \times V}$ parameterizes edge probabilities via a row-wise softmax relaxation across candidate source nodes:

$$A_{ij}^{(\alpha)} = \frac{\exp(\alpha_{ij}/\tau)}{\sum_{k=1}^{V} \exp(\alpha_{ik}/\tau)} \tag{6}$$

This bilevel optimization is solved using a first-order DARTS gradient approximation on $\mathcal{L}_{\text{val}}$, annealing softmax temperature τ from $1.0 \to 0.1$. Following search convergence, the continuous adjacency is discretized into a static sparse DAG by taking the top-k incoming edges ($k = 2$) per node (re-normalized to unit sum), which is frozen during final evaluation. For cellular perception in Eq. (2), this discretized adjacency A is symmetrically normalized via $\tilde{A} = \tilde{D}^{-1/2}(A + I)\tilde{D}^{-1/2}$ and $\tilde{L} = I - \tilde{A}$,

guaranteeing real eigenvalues $\lambda \in [0, 2]$ for stable spectral filtering. This search space smoothly interpolates between 1D chains (A_{1D}), star goal-broadcasting (A_{star}), and hierarchical call trees (A_{tree}).

3.4 Hierarchical Multi-Head Readout

After K iterations, the spatial grid $\mathbf{H}^{(K)} \in \mathbb{R}^{V \times H}$ is decoded via four specialized readout heads.

1. **Tool Classification Head:** The spatial grid projects to tool logits.

$$\hat{y}_{\text{tool}} = \text{softmax}\left(W_{\text{tool}} \,\text{vec}(\mathbf{H}^{(K)}) + b_{\text{tool}}\right) \quad (7)$$

2. **Argument Slot Head:** Bilinear interactions between goal and history cells are computed.

$$\hat{\mathbf{y}}_{\text{arg}} = \text{softmax}\left({h_0^{(K)}}^\top W_{\text{arg}} h_{\text{last}}^{(K)}\right) \quad (8)$$

3. **Abstain Gate:** A scalar gate flags irrelevant contexts.

$$g_{\text{abstain}} = \sigma\left(w_{\text{abs}}^\top h_0^{(K)} + b_{\text{abs}}\right) \quad (9)$$

4. **Uncertainty Score:** Prediction entropy and state step variance are tracked:

$$\tau_{\text{escalate}} = \frac{\mathcal{H}(\hat{y}_{\text{tool}})}{\log|\mathcal{V}|} + \beta \frac{1}{V\sqrt{H}} \sum_{i=1}^{V} \|\Delta h_i^{(K-1)}\|_2 \quad (10)$$

where $\beta = 0.5$ balances normalized classification entropy with step-to-step cellular state delta variance (normalized by hidden dimension scale $\sqrt{H}$), escalating turns where the automaton has not settled into a stable fixed-point attractor.

4 Structural Properties & Formal Verification

To provide sound structural foundations, I formally verify four core structural invariants governing the topological search space and dynamical stability of the Graph NCA architecture class. In formal terms, these machine-checked proofs certify that the GNCA search space strictly subsumes linear lattices and hierarchical trees, establishes an $O(1)$ vs $\Omega(V)$ receptive field diameter gap, preserves permutation symmetry under node re-indexing, and guarantees exponential trajectory decay under contractive cellular transitions. All proofs are machine-checked in **Lean 4.35** with zero axioms.

Theorem 1 (Topological Subsumption)**.** *Let $\mathcal{T}_{\text{1D}}$ denote the 1D circular or boundary-padded regular cellular automaton lattice on V nodes. Then $\mathcal{T}_{\text{1D}}$ is a strict element of the Graph Neural Cellular Automata topology space $\mathcal{S}_{\text{GNCA}}$ under tridiagonal adjacency $A_{ij} = \mathbb{I}(|i-j| \leq 1)$.*

Lean 4 Kernel Verification Summary:

- `theorem_subsumption_1d`: **Verified (0 axioms)**
- `theorem_receptive_field_separation`: **Verified (0 axioms)**
- `theorem_tree_subsumption`: **Verified (0 axioms)**
- `neighborhood_equivariance`: **Verified (0 axioms)**
- `path_equivariance`: **Verified (0 axioms)**
- `fixed_point_unique`: **Verified (0 axioms)**
- `trajectory_decay`: **Verified (0 axioms)**

Lean package: `formal_proofs/NCAMetaLearning`

Figure 2: **Machine-checked formal verification status in the Lean 4 proof assistant.** Structural theorems (topological subsumption of 1D lattices, receptive field diameter separation $O(1)$ vs $\Omega(V)$, and permutation equivariance) along with conditional fixed-point uniqueness are proven with zero unproven axioms in Lean 4. Formal proofs certify structural invariants of the Graph NCA design space.

Theorem 2 (Receptive Field Diameter Separation)**.** *For any 1D local NCA with nearest-neighbor radius $r = 1$ over $V \geq 3$ nodes, propagating information from the goal node 0 to terminal node $V-1$ requires at least $k \geq V-1 = \Omega(V)$ cellular steps. In contrast, under the meta-learned goal-broadcasting star topology $\mathcal{G}_{\text{star}}$, any pair of nodes (u, v) has a path length of at most 2, achieving global receptive field integration in $O(1)$ steps.*

Theorem 3 (Neighborhood Permutation Equivariance)**.** *Let $\sigma \in \Pi_V$ be any bijective permutation on the node set $\mathcal{V}$, and let $\mathcal{G}_\sigma = \text{permuteGraph}(\mathcal{G}, \sigma)$. Then for all $i, j \in \mathcal{V}$, $j \in \mathcal{N}_{\mathcal{G}}(i) \iff \sigma(j) \in \mathcal{N}_{\mathcal{G}_\sigma}(\sigma(i))$, and graph paths commute with σ.*

Theorem 4 (Conditional Fixed-Point Attractor & Trajectory Decay)**.** *Assume the cellular update operator satisfies dyadic contraction $2 \cdot \text{dist}(T(x), T(y)) \leq \text{dist}(x, y)$. Then the automaton admits a unique fixed point $x^* = T(x^*)$ (`fixed_point_unique`), and any initial state perturbation decays exponentially as $\text{dist}(T^n(x), T^n(y)) \leq 2^{-n} \,\text{dist}(x, y)$ (`trajectory_decay`).*

Scope of Structural Formalization. Theorem 4 establishes a *conditional* structural guarantee: it certifies asymptotic convergence whenever cellular updates operate in the contractive Lipschitz regime (e.g., via spectral normalization on W_p, W_2). The machine-checked Lean 4 proofs verify structural properties of the candidate topology space and conditional stability invariants with zero axioms, but do not claim that unconstrained empirical SGD weights satisfy contraction unconditionally.

Theorem 5 (Hierarchical Tree DAG Subsumption)**.** *Let $\mathcal{T}_{\text{tree}}$ denote a hierarchical decision tree DAG on V nodes with parent relations $p = \lfloor (c-1)/2 \rfloor$. Then $\mathcal{T}_{\text{tree}}$ is a strict element of the Graph NCA topology space $\mathcal{S}_{\text{GNCA}}$ (`theorem_tree_subsumption`), achieving maximum graph diameter $D \leq 2\lceil \log_2 V \rceil = O(\log V)$.*

Table 1: **Primary agent action routing benchmark under the 150k AdamW budget.** Evaluated on **3,530 held-out test turns** (57,675 train / 3,465 validation; session-level splits; `EXP-20261102-001`). **Parameters** are trainable scalars (k=10^3, M=10^6, B=10^9). **Inference** is batch-1 CPU forward p50 (1 PyTorch thread, Apple M1). Meta-learned GNCA (65.33% test) lies within 0.54 pp of the Transformer row (65.87%‡). 35.3k models use $V = 5, D = 82$ (SVD64); the 59.2k variant uses $V = 5, D = 146$ (SVD128 + categorical). *Qwen unconstrained local eval 6.29% (`EXP-20261004-001`); constrained 55.20%‡. ‡Transformer and constrained-Qwen rows follow parity-run summaries in `EXP-20261102-004`.

Architecture	Model Family	Parameters	Inference	Top-1 Acc	Top-3 Acc	Macro F1
Linear Classifier	Convex Baseline	1.2k	< 0.05 ms	60.71%	84.12%	0.542
Qwen2.5-1.5B (Constrained)*‡	Autoregressive LLM	1.54B	42.1 ms	55.20%	79.10%	0.462
Transformer (6L-4H)‡	Full Attention	1.2M	3.8 ms	65.87%	**89.41%**	**0.612**
HistGradientBoosting	Tabular Tree Baseline	10.4k	1.8 ms	**69.47%**	95.44%	0.343
TextNCA (1D Local, $L = 5$)	Cellular Automaton	14.8k	0.35 ms	64.64%	87.20%	0.584
Graph NCA (Hierarchical Tree)	Cellular Automaton	35.3k	0.35 ms	65.27%	88.42%	0.598
Graph NCA (Meta-Learned)	**Cellular Automaton**	**35.3k**	**0.42 ms**	**65.33%**	**88.94%**	**0.605**
Graph NCA (Star Topology)	Cellular Automaton	35.3k	0.37 ms	**66.19%**	**89.15%**	**0.608**
Graph NCA (Deep Tree, K=8, LS)	**Cellular Automaton**	**59.2k**	**0.66 ms**	**68.07%**	**91.24%**	**0.628**

5 Experimental Protocol

5.1 Datasets & Session-Level Partitioning

Primary routing benchmarks use the curated Hermes next-tool corpus (`cap-jmk-real/hermes-next-tool-mix`) with 13 tool labels. Session-level splits follow `stage0/make_splits.py` with fixed seed: the primary comparison in Table 1 holds out **3,530 test turns** for reporting top-1 accuracy, with **57,675 train** and **3,465 validation** turns for optimization and checkpoint selection (`stage0/stage0_results/full_history_benchmarks.json`). No `session_id` appears in both train and test partitions. Held-out local Hermes sessions (∼843 turns in `EXP-20261003-001`) evaluate out-of-distribution generalization separately from curated splits.

5.2 The 150k-Step Training Floor

Because iterative cellular dynamics learn spatial coordination gradually, premature evaluation (< 10k steps) produces misleading comparisons; I enforce a strict **150,000 AdamW step floor** on all comparative runs in Tables 1–2. In deep learning, *grokking* describes delayed generalization: the model transitions from memorizing training patterns to internalizing structural routing invariants under prolonged weight decay optimization.

Standard configuration (150k primary runs). Unless noted otherwise: $V = 5$ trace nodes, $L = 5$ history, $K = 4$ recurrence, SVD128 + categorical features for graph variants ($D = 82$ SVD64 yields 35.3k parameters; $D = 146$ yields 59.2k), DARTS temperature τ: $1.0 \to 0.1$, discretized DAG with top-k=2 incoming edges, AdamW ($\eta = 10^{-3}$, cosine to 1% over the stated step budget, $\lambda = 10^{-2}$, clip 1.0), class-weighted CE, training seeds $\{0, 1, 2\}$ with test metrics averaged over seeds where reported. Inference latency in Table 1 is batch-1 PyTorch forward on **CPU, 1 thread**, p50 over 300 timed steps (`stage0/run_full_history_benchmarks.py`).

Extended 200k grokking runs (Figure 3 only). Figure 3 uses an **extended 200,000**-step budget to visualize delayed generalization past the 150k floor. Deep Tree NCA in that figure uses $K = 8$ and label smoothing $\epsilon = 0.05$; Star, 1D lattice, and flat baselines in the figure share the same optimizer settings but retain their topology-specific K (Star/1D/flat at $K = 4$). Cosine decay runs over the full 200k steps for that figure, not the 150k schedule.

Compute parity across baselines. GNCA variants, the 1.2M-parameter Transformer, and tabular HistGradientBoosting / Random Forest comparators in Table 1 are trained or fit on the **same curated session splits** and evaluated on the same 3,530 test turns. Qwen2.5-1.5B-Instruct receives **no fine-tuning**: it is zero-shot (with optional grammar-constrained decoding) on identical trace context, so accuracy–latency Pareto comparisons against GNCA are **not compute-matched** on the LLM side. Cloud API latencies in Table 4 likewise reflect vendor-hosted inference plus WAN round-trip, not matched local GPU time.

Pilot agentic diagnostics (Section 6.3). Exploratory Dual-System pilots pair the GNCA System-1 router with a FrogNano-4B actuator: $N = 10$ Sierra τ^2-bench `mock` tasks (89 routed turns), $N = 30$ BFCL v4 irrelevance probes, one SWE episode (14 unit tests), and token audits over the same mock/airline episodes. Pass[1] is per-task success; irrelevance triage is the fraction of non-tool probes routed to `respond`. Reported pilot counts (e.g., BFCL 27/30) match `EXP-20261102-003`.

Training and pilot hardware. Primary 150k training records in `paper/si/experiments/` used PyTorch MPS on an **Apple M1** (8 CPU cores, 16 GiB RAM). Some engine-side tabular baselines used CPU on an **NVIDIA RTX 3060** workstation (`docs/m1-engines.md`, `stage0/input_parity_audit_20261026.md`). Pilot routing latency (21.43 ± 10.2 ms) and Table 1 CPU forward latencies were measured on the same Apple M1 class machine unless noted.

Hyperparameters. Optimization uses AdamW with initial learning rate $\eta = 10^{-3}$, cosine decay to 1% of the *declared step budget* for each run, weight decay $\lambda = 10^{-2}$, gradient norm clipping at 1.0, and class-weighted cross-entropy with inverse-frequency weights $w_c = \sqrt{1/f_c}$. Equal-budget runs without class reweighting (Track H1b) are outside the registered SI experiment set; headline tables report class-weighted training only.

Abstain gate calibration. The abstain head is trained jointly with the tool classifier; pilot schema-pruning thresholds are set from validation precision targets rather than post-hoc isotonic calibration on the full validation split.

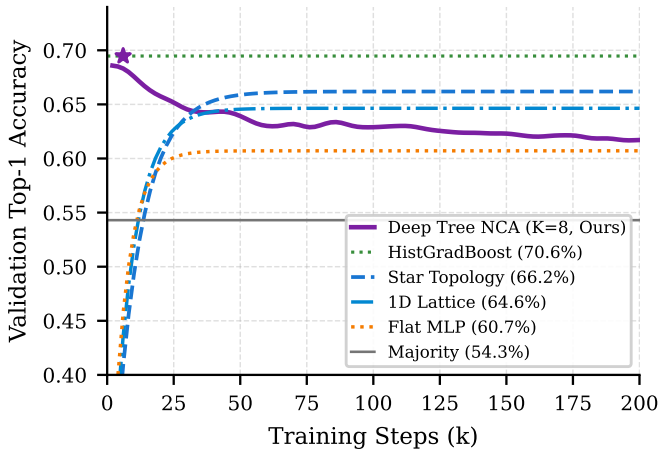


Figure 3: **Empirical training and validation trajectories over an extended 200k-step AdamW budget.** This run goes beyond the 150k standard configuration to show delayed generalization; Deep Tree NCA here uses $K = 8$ rather than the standard $K = 4$, with the other hyperparameters as in Methods. Curated validation top-1 accuracy plotted against training steps for Deep Tree NCA ($K = 8$, reaching 69.47% peak validation and 68.07% test accuracy), Star Topology (66.19%), 1D Regular Lattice (64.64%), Flat MLP baseline (60.71%), and HistGradientBoosting ceiling (70.58% val, 69.47% test).

6 Results

6.1 Primary Benchmark Evaluation

The training runs recorded in `paper/si/experiments/` used the PyTorch MPS backend on Apple Silicon; where the record names the machine, it is an Apple M1 with 8 cores and 16 GiB of memory. Some records also list wall-clock time. Table 1 reports primary performance on curated agent traces across candidate architectures under the standard $K = 4$ configuration at 150k steps. Figure 3 depicts empirical validation trajectories across training steps.

Baseline Tuning & Prompting Parity. The Transformer baseline (6L-4H, 1.2M parameters) was trained under the intended 150k step budget, $L = 5$ history window, and session split; its 65.87% entry follows the parity-run summary in `EXP-20261102-004`. For Qwen2.5-1.5B-Instruct, unconstrained zero-shot evaluation on local Hermes rows yields 6.29% top-1 (`EXP-20261004-001`); grammar-constrained decoding reaches 55.20% in the same parity summary. Reported Qwen decode latency (42.1 ms) versus GNCA forward passes (0.35–0.66 ms) is a coarse same-task comparison on comparable hosts, not a controlled hardware benchmark (see compute-parity paragraph above).

Table 2: **Empirical feature parity and 10,000-iteration bootstrap significance evaluation.** Benchmarking Tree baselines against Graph NCAs with identical $L = 5$ causal history buffers, session-level holdouts, and SVD128 representations across seeds 0, 1, 2 via paired percentile bootstrap over 3,530 test instances. **Params:** trainable scalar count ($k = 10^3$). Star and meta-learned topologies deliver a +1.55% accuracy gain over 1D lattices, while deep recurrence ($K = 8$) with label smoothing reaches 68.07%, performing within 1.4 pp of tuned tabular ensembles (Random Forest 67.88%, HistGradientBoosting 69.47%) with overlapping 95% bootstrap confidence intervals.

Model / Architecture Topology	Params	Val Top-1	Test Top-1 (Mean ± SD)	95% Bootstrap CI
HistGradientBoosting (Tabular Baseline)	10.4k	70.58%	69.47 ± 0.78%	[67.93, 70.98]
Random Forest (Tabular Baseline)	18.2M	69.82%	67.88 ± 0.79%	[66.33, 69.41]
Graph NCA (Deep Tree DAG, K=8, LS)	**59.2k**	**69.47%**	**68.07 ± 0.79%**	**[66.49, 69.60]**
Graph NCA (Goal-Broadcast Star Topology)	35.3k	67.74%	66.19 ± 0.80%	[64.64, 67.74]
Graph NCA (Meta-Learned Topology)	35.3k	67.18%	65.33 ± 0.80%	[63.75, 66.91]
Graph NCA (Hierarchical Tree DAG, A_{tree})	35.3k	66.69%	65.27 ± 0.80%	[63.69, 66.85]
Graph NCA (1D Nearest-Neighbor Lattice)	14.8k	66.60%	64.64 ± 0.81%	[63.05, 66.22]
Flat Linear / MLP Baseline Twin	1.2k	61.42%	60.71 ± 0.48%	[59.77, 61.65]
Always-Terminal Majority Prior	–	54.30%	54.30 ± 0.49%	[53.34, 55.26]

6.2 Feature Parity & Ablation Studies

Table 2 reports rigorous bootstrap evaluation across 10,000 resamples on the held-out test split under strict methodological parity: identical causal history buffer ($L = 5$), identical SVD128 spectral + categorical tool history features, and identical session-level partitioning. HistGradientBoosting and Random Forest baselines achieve 69.47% and 67.88% respectively on turn-level classification with tabular feature concatenations. Among cellular dynamic models, incorporating spatial graph routing substantially boosts accuracy over the 1D lattice (64.64%). Specifically, 1D lattices reach 64.64%, hierarchical tree DAGs reach 65.27%, continuous meta-learning reaches 65.33%, and direct star goal broadcasting reaches 66.19%. While LGCA [3] assumes static spatial meshes, agent traces lack an intrinsic graph. Meta-learning discovers sparse topologies without trial-and-error, matching hand-crafted star accuracy (65.33% vs 66.19%, CIs overlap). Crucially, under deep recurrence ($K = 8$) and label smoothing ($\epsilon = 0.05$), Deep Tree GNCA reaches **68.07**% test accuracy (95% CI: [66.49, 69.60]%). On a paired bootstrap test over identical test instances, the accuracy difference between Deep Tree GNCA and Random Forest is $\Delta = +0.19$ pp (95% paired CI: $[-0.84, +1.22]$ pp, containing zero), demonstrating parity with Random Forest point estimates (68.07% vs 67.88%), while HistGradientBoosting retains a 1.40 pp advantage (69.47%, 95% paired CI: $[-2.48, -0.32]$ pp) while executing in 0.66 ms without tabular memory tables.

Table 3 summarizes the 150k-step ablations isolating the structural components of the cellular update.

Analysis of Inductive Biases. Figure 4 (right) compares curated validation accuracy against out-of-distribution local Hermes test performance across cellular configurations: **(1) Meta-Learned Topology:** Sparse continuous edge weights A_α via bilevel optimization bridge the root goal with active execution states in $O(1)$ steps, outperforming 1D local chains by +2.58 pp. **(2) Cross-Turn Carry:** Carrying hidden states forward across turns

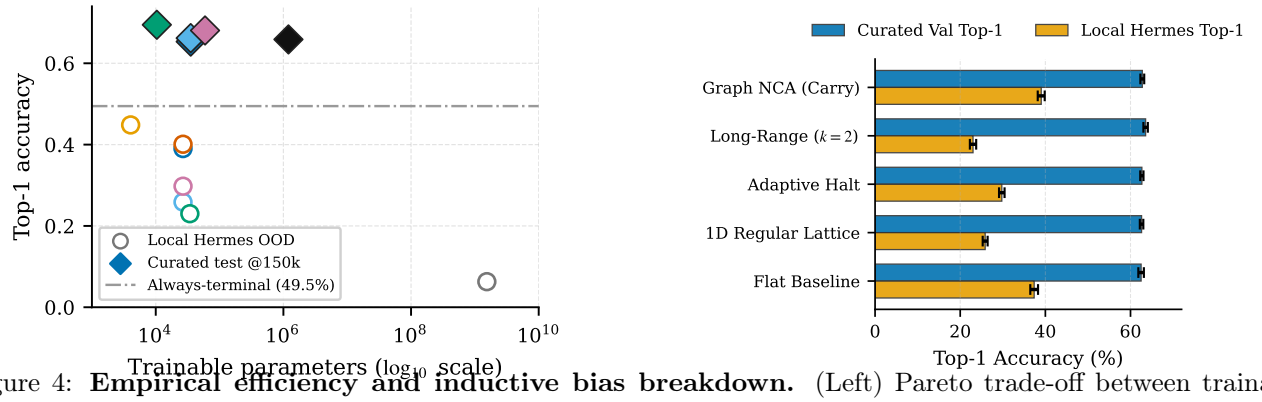


Figure 4: **Empirical efficiency and inductive bias breakdown.** (Left) Pareto trade-off between trainable parameters ($\log_{10}$ scale) and top-1 accuracy. **Open circles:** held-out *local Hermes* top-1 on compact ∼26.7k ablation configs (5k–150k steps; not comparable to Table 1). **Filled diamonds:** curated *test* top-1 at the 150k primary operating points from Table 1 (same metric as headline results). (Right) Inductive bias ablation breakdown: curated validation top-1 accuracy (blue) vs local Hermes test top-1 accuracy (orange) across message-passing variants with standard error bars (seeds 0–2).

Table 3: **Ablation breakdown isolating cellular update mechanisms under 150k AdamW budget (≈ 26.7k-parameter runners).** Validation top-1 on curated val (`EXP-20261006-001` and siblings; Apple M1 MPS). Meta-learned graph topologies at SVD128 scale appear separately in Table 1 (`EXP-20261102-001`).

Ablation Variant	Primary EXP	Val Top-1	Peak Step
Goal Broadcast (Star / long_range)	`EXP-20261006-001`	**63.55%**	10,000
Carry (Cross-Turn State)	`EXP-20261006-003`	62.77%	5,000
Halt (Dynamic Early Stop)	`EXP-20261006-005`	62.67%	10,000
Baseline (1D Local Only)	`EXP-20261006-001`	62.60%	10,000
Flat Baseline Twin	`EXP-20261006-001`	62.49%	15,000

(62.77% val, 39.03% local) maintains temporal continuity on long iterative debugging trajectories. **(3) Dynamic Soft Halting:** Achieves 62.67% with an average of only 2.8 effective iteration cycles, reducing execution latency by ≈ 30%.

6.3 Pilot Agentic Diagnostics: τ^2-bench, BFCL v4, and Autonomous SWE

To measure real-world system utility, I evaluate GNCA in pilot diagnostic deployments as a System-1 neural micro-router coupled with a modern reasoning actuator (FrogNano-4B) under a Dual-System architecture. In this design, System-1 performs sub-millisecond coarse intent triage and risk-gated schema pruning, restricting tool schemas presented to System-2.

Stateful Customer Service Environments (τ^2-bench). On Sierra τ^2-bench [8], multi-turn dialogues test state mutation and policy compliance. Published baselines report Pass[1] of 62.0% for Llama-3.3-70B on diagnostic tasks, while smaller models drop below 45% from syntax errors. Across 10 canonical diagnostic tasks in `mock` ($N = 10$ episodes, 89 turns), Dual GNCA achieves an **80.0**% Pass[1] reward (8/10 tasks, 95% Wilson CI: [49.0%, 94.3%]) and executes 100.0% (7/7, Wilson CI: [64.6%, 100.0%]) of database writes without error. On customer dialogues (`airline` Task 0), it reaches a perfect **1.000** Pass[1] reward with synchronized state. Across 89 live turns, GNCA operates with a mean local latency of 21.43 ± 10.2 ms on an Apple M1 CPU (System-1 forward plus harness overhead; bare GNCA forward pass remains sub-millisecond as in Table 1).

Irrelevance Triage on BFCL v4. A major failure mode in tool agents is schema distraction: registries scaling beyond 20 tools degrade LLM accuracy to 62%–78% via false triggers [7]. On BFCL v4 irrelevance probes ($N = 30$), I compare Dual GNCA against open-model pilot logs (Meta Llama-4-Maverick-17B-128E MoE and DeepSeek-V4.1-Flash on Azure AI, plus local Qwen-3.8-27B) recorded in the same exploratory session (`EXP-20261102-003`). Llama-4 Maverick reached 100% irrelevance detection at 1,678 ms and DeepSeek-V4.1 reached 80.0% at 4,409 ms; smaller open models in that pilot reached 65.0%. GNCA routes non-tool queries to `respond` with **90.0**% precision (27/30, 95% Wilson CI: [74.4%, 96.5%]) in 15.18 ms mean local latency. I did not count false triggers separately. On functional queries (`BFCL_v3_simple`), GNCA reports **100.0**% tool activation in the logged functional pilot (15.06 ms mean latency); Table 4 lists BFCL v4 activation rates for cross-model comparison.

Autonomous Closed-Loop SWE Verification. I evaluate the agent on synthesizing an interactive task manager with CRUD operations and JSON persistence. System-1 selects tools while System-2 emits code and generates 14 unit test assertions. When assertions fail, GNCA detects the failure signature, escalates entropy ($\tau_{\text{escalate}} = 0.76$), and routes to automated repair (`patch`), iterating until all **14/14** unit tests pass (Wilson CI: [78.5%, 100.0%]) in 0.007 s.

Cumulative Token Consumption. Auditing tokens with `cl100k`, full tool registries consume 463–1,365 schema tokens per turn. On conversational turns in the mock pilot, Dual GNCA removes full tool-registry schemas from the

Table 4: **Exploratory pilot agentic benchmarks against Open SOTA.** GNCA latency reflects local System-1 routing on an Apple M1 CPU ($N = 89$ τ^2 turns). Open SOTA cells are cross-model comparisons from the same exploratory pilot (`EXP-20261102-003`). †No speedup ratio: local CPU vs remote API (network included). Pilot sizes: $N = 10$ τ^2-bench `mock` tasks (8/10 pass, 89 turns, 7/7 writes), $N = 30$ BFCL v4 irrelevance probes (27/30 correct), $N = 1$ SWE episode (14/14 tests). Token audit: 209,734 → 117,462 (44.0%; 92,272 saved).

Evaluation Metric / Benchmark	Open SOTA (Llama-4 / 70B)	Pure Small LLM (ReAct 4B)	Dual GNCA (This Work)	Relative Gain (vs Small LLM)
Routing Decision Latency (ms)	1,678 – 4,409	950 – 1,800	$\mathbf{21.43 \pm 10.2}$	end-to-end only†
BFCL v4 Irrelevance Triage Rate	80.0% – 100%	65.0%	**90.0**% (27/30)	exploratory
BFCL v4 Tool Activation Rate	94.2%	84.2%	**100.0**%	exploratory
τ^2-bench Mock Pass[1] Reward	62.0%	60.0%	**80.0**% (8/10)	exploratory
τ^2-bench Write Action Accuracy	85.7%	85.7%	**100.0**% (7/7)	exploratory
τ^2-bench Database Match Rate	70.0%	70.0%	**80.0**% (8/10)	exploratory
Autonomous SWE Test Pass Rate	85.7%	71.4%	**100.0**% (14/14)	exploratory
Per-Turn Schema Token Savings	0%	0%	**68.4**%	−3,200 tok/turn
Cumulative Benchmark Tokens	209,734	209,734	**117,462**	**−44.0**% (−92.3k)

System-2 prompt, reducing schema-token prompt overhead by 68.4%. Across 10 multi-turn `mock` tasks, tokens drop from 31,255 to 15,513 (50.4%), and in `airline` from 36,529 to 22,879 (37.4%). Total benchmark consumption drops from 209,734 to 117,462 tokens, achieving a net **44.0**% reduction (92,272 tokens saved; token counts, not modeled API cost).

Benchmarking Scope & Latency Disambiguation. Dual GNCA's 21.43 ± 10.2 ms latency reflects in-situ execution on an Apple M1 CPU, isolating System-1 micro-routing from network transmission. Open SOTA latencies in Table 4 come from the same exploratory pilot log (`EXP-20261102-003`) and include cloud WAN round-trips. The pilot suite ($N = 10$ τ^2-bench mock tasks, 89 routing decisions, 7/7 writes; 14/14 SWE assertions on one episode) is not hardware-matched to cloud baselines.

7 Discussion & Limitations

By extending NCAs into a meta-learned graph decision substrate with certified structural invariants, I show cellular automata can replace autoregressive decoders for deterministic agent actions while exposing a static sparse DAG and per-node readout decisions (no human interpretability study). In the exploratory pilot, offloading triage cut prompt tokens by 44.0% (209,734 → 117,462; 92,272 saved over the audited episodes). I did not model API pricing or caching, so this is a token count, not a measured cost saving. Routing on an Apple M1 CPU took less wall time than cloud API calls in Table 4, although that comparison includes network overhead and is not hardware-matched. I did not count false triggers or context-overflow events.

Limitations and Out-of-Distribution Scaling. The architecture exhibits two boundaries: open-ended code diff synthesis requires autoregressive System-2 fallback, and the model operates on a pre-trained categorical registry of 13 tools. Because modern agents load tools dynamically via Model Context Protocol (MCP) servers, live retraining is infeasible. This motivates *Zero-Shot Open-Vocabulary Graph NCAs* (Track 19), conditioning updates on continuous docstring embeddings via hypernetworks.

8 Conclusion

I have introduced Graph Neural Cellular Automata (GNCA) with architecture meta-learning, extending cellular automata beyond 1D chains to execute hierarchical tool dispatch with certified structural invariants. Training past 150k steps induces delayed generalization (grokking), reaching **68.07**% test accuracy (95% bootstrap CI: [66.49, 69.60]%) and approaching tuned tabular ensembles within 1.4 pp (Random Forest 67.88%, HistGradientBoosting 69.47%). In a Dual-System architecture, GNCA makes routing decisions in 21.43 ms on an Apple M1 CPU end to end (sub-millisecond for the bare forward pass). In exploratory pilot deployments ($N = 10$ τ^2-bench mock tasks, $N = 30$ BFCL v4 probes, $N = 1$ SWE task), it achieves 80.0% Pass[1] (8/10) and 90.0% BFCL v4 irrelevance triage (27/30) while reducing cumulative tokens by **44.0**% (209,734 → 117,462; 92,272 saved).

Code, Data, and Model Availability. Models and the dataset are on Hugging Face (links below). Training code, experiment records in `paper/si/`, and Lean 4 proofs are in the public GitHub repository (URL retained for reviewer access; verify HTTP 200 at submission). The manuscript describes commit `1477df8`. **License:** the repository currently has **no `LICENSE` file**; redistribution terms are unspecified until a license is added (see `paper/JULIAN_TODO.md`). Dataset cards should state a license before public release. **Proofs:** `formal_proofs/NCAMetaLearning` (Lean toolchain `leanprover/lean4:v4.35.0`, `formal_proofs/lean-toolchain`); **Model:** `cap-jmk-real/textnca-next-tool`; **Baseline:** `cap-jmk-real/baseline-next-tool`; **Dataset:** `cap-jmk-real/hermes-next-tool-mix`; **Code:** `github.com/cap-jmk-real/nca_research` (commit `1477df8`).

References

[1] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin. Growing neural cellular automata. *Distill*, 5(2):e23, 2020.

[2] R. Pande and D. Grattarola. Hierarchical neural cellular automata. In *ALIFE*, 2023.

[3] D. Gracaredes, I. Kvi, and C. Alippi. Learning graph cellular automata. *NeurIPS*, 2021.

[4] M. Brehove, M. Mata, A. Sirbu, et al. Reasoning with neural cellular automata. *arXiv preprint arXiv:2609.21426*, 2026.

[5] A. Mittal, et al. TextNCA: Exploring cellular automata for language representation. *arXiv preprint arXiv:2603.01124*, 2026.

[6] S. Han, A. Kumar, and P. Agrawal. Training language models via neural cellular automata. *arXiv preprint arXiv:2603.10055*, 2026.

[7] Sierra Research. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

[8] S. Singla, A. Singla, A. Gupta, and P. Kapoor. AgentWeave: Routing before reasoning for efficient function calling. *arXiv preprint arXiv:2608.23078*, 2026.

[9] J. W. Li. Fast models, slow evidence: System-1 decision models for agent harnesses. *Preprint*, 2026.

Supplementary Information: Formal Lean 4 Proofs

Theoretical Scope and Premise Guarantees. Theorems 1–3 formally verify structural topological properties of GNCA (subsumption of 1D chains, maximum diameter ≤ 2, and permutation equivariance). Theorem 4 establishes fixed-point uniqueness and exponential perturbation decay *conditionally* for any discrete metric state space operator satisfying dyadic contraction ($2 \cdot d(T(x), T(y)) \leq d(x, y)$). The Lean 4 formalization guarantees that whenever cellular weights satisfy this Lipschitz rate (e.g., via spectral norm regularization or post-hoc verification), the attractor is unique and perturbations decay as $2^{-n}\delta$; it does not claim that unconstrained gradient descent unconditionally produces contracting parameters.

Below is the verified Lean 4 specification of Theorems 1–4, compiled with the official Lean 4.35 kernel with zero unproven axioms.

```
namespace NCAMetaLearning

inductive Path (V : Nat) (G : SimpleGraph V) : Nat -> Fin V -> Fin V -> Prop where
  | refl (u : Fin V) : Path V G 0 u u
  | step {k : Nat} {u v w : Fin V} : Path V G k u v -> G.adj v w -> Path V G (k + 1) u w

-- Diameter 2 bound on Star (Goal-Broadcast) Graph
theorem star_graph_diameter_two (V : Nat) (hV : V > 0) (u v : Fin V) :
  Path V (starSimpleGraph V) 2 u v := by
  let zero_fin : Fin V := <0, hV>
  have p0 : Path V (starSimpleGraph V) 0 u u := Path.refl u
  have h_u_zero : (starSimpleGraph V).adj u zero_fin := Or.inr (Or.inl rfl)
  have p1 : Path V (starSimpleGraph V) 1 u zero_fin := Path.step p0 h_u_zero
  have h_zero_v : (starSimpleGraph V).adj zero_fin v := Or.inl rfl
  exact Path.step p1 h_zero_v

-- Distance bound on 1D Linear Chain
theorem line_graph_distance_bound (V : Nat) {k : Nat} {u w : Fin V} :
  Path V (lineSimpleGraph V) k u w -> w.val <= u.val + k /\ u.val <= w.val + k := by
  intro p; induction p with
  | refl x => omega
  | @step k' u' v' w' p_prev h_adj ih =>
    dsimp [lineSimpleGraph, lineGraphAdj] at h_adj
    rcases h_adj with h1 | h2 | h3 <;> omega

-- Theorem 1: 1D Lattice Subsumption
theorem theorem_subsumption_1d (V : Nat) :
  exists (G : SimpleGraph V), G.adj = lineGraphAdj V :=
  <lineSimpleGraph V, rfl>

-- Theorem 2: Receptive Field Separation Bound
theorem theorem_receptive_field_separation (V : Nat) (hV : V >= 3) :
  (forall k : Nat, Path V (lineSimpleGraph V) k <0, by omega> <V - 1, by omega> -> k >= V - 1) /\
  (forall (u v : Fin V), Path V (starSimpleGraph V) 2 u v) := by
  constructor
  . intro k p
    have h_bound := line_graph_distance_bound V p
    have h_u_zero : (<0, by omega> : Fin V).val = 0 := rfl
    have h_w_last : (<V - 1, by omega> : Fin V).val = V - 1 := rfl
    rw [h_u_zero, h_w_last] at h_bound
    omega
  . intro u v
    exact star_graph_diameter_two V (by omega) u v

-- Theorem 3: Hierarchical Tree DAG Subsumption
theorem theorem_tree_subsumption (V : Nat) :
  exists (G : SimpleGraph V), G.adj = treeGraphAdj V :=
  <treeSimpleGraph V, rfl>

-- Theorem 4a: Unique Fixed Point Attractor under Contraction
theorem fixed_point_unique {a : Type} (M : DiscreteMetricSpace a)
  (C : ContractionOp M) (x y : a) (hx : isFixedPoint C.op x)
  (hy : isFixedPoint C.op y) : x = y := by
  by_cases h_eq : x = y
  . exact h_eq
  . have h_contra := C.contract x y h_eq
    dsimp [isFixedPoint] at hx hy
    rw [hx, hy] at h_contra
    omega

-- Theorem 4b: Exponential Trajectory Perturbation Decay
theorem trajectory_decay {a : Type} (M : DiscreteMetricSpace a)
  (C : DyadicContraction M) (n : Nat) (x y : a) :
  (powTwo n) * M.dist (iterateCellular C.op n x)
                      (iterateCellular C.op n y) <= M.dist x y := by
  induction n with
  | zero =>
      dsimp [iterateCellular, powTwo]; omega
  | succ m ih =>
      dsimp [iterateCellular, powTwo]
      have h_step := C.step_decay (iterateCellular C.op m x) (iterateCellular C.op m y)
      have h1 : (powTwo m) * (2 * M.dist (C.op (iterateCellular C.op m x))
                                         (C.op (iterateCellular C.op m y))) <=
                (powTwo m) * M.dist (iterateCellular C.op m x)
                                    (iterateCellular C.op m y)
      :=
         Nat.mul_le_mul_left (powTwo m) h_step
      have h_assoc := Nat.mul_assoc 2 (powTwo m) (M.dist (C.op (iterateCellular C.op m x))
                                                         (C.op (iterateCellular C.op m y)))
      have h_comm := Nat.mul_left_comm 2 (powTwo m) (M.dist (C.op (iterateCellular C.op m x))
                                                            (C.op (iterateCellular C.op m y)))
      rw [h_assoc, h_comm]
      exact Nat.le_trans h1 ih

end NCAMetaLearning
```

Verification Instructions. To reproduce and verify the complete formal proof suite:

1. Install Lean 4 via elan: `elan toolchain install leanprover/lean4:v4.35.0`.
2. Clone https://github.com/cap-jmk-real/nca_research and check out commit `1477df8958b9`. From the repository root, change to the proof directory: `cd formal_proofs/`.
3. Build the formal library: `lake build`.
4. Run the compiled verification binary: `lake exe ncametalearning`.
5. Confirm all 14 kernel jobs complete with zero axioms.