Reasoning: When Euler Meets Stack

Computational Boundaries, Incompleteness, and the Necessity of Discrete Dynamics

Zixi Li
Independent Researcher
lizx93@mail2.sysu.edu.cn

November 28, 2025

Abstract

We present a fundamental critique of contemporary deep learning approaches to reasoning, grounded not in empirical failure but in *categorical necessity*. Our central thesis unfolds in three parts:

Part I (The Problem): We prove that all sequential models—Transformers, RNNs, and their variants—are structurally incapable of reasoning. This failure is *not* due to insufficient representation capacity: modern floating-point systems (BF16/FP32) already provide state spaces orders of magnitude larger than required for planning, game-playing, and theorem-proving tasks. The failure stems from **operator category mismatch**—attempting to model reasoning with pseudo-Euclidean dynamics that inevitably collapse into irreversible, semantically lossy RNN-like structures.

Part II (Ignored Reality): Drawing on recent Monte Carlo experiments [1], we establish that **computational boundaries exist** as sharp phase transitions, not merely as asymptotic complexity classes. Furthermore, building on incompleteness theory [2], we show that reasoning systems cannot be complete without prior anchors. Yet these boundaries are not Lipschitz-contraction guarantees—they are *information-theoretic phase transitions* with measurable critical densities.

Part III (The Solution): We introduce stack-based reasoning systems with computational boundaries and prove the **Euler-Stack Correspondence Theorem**: pointer dynamics in bounded stack spaces are isomorphic to *honest discrete Euler iterations* with guaranteed convergence. Crucially, we show that **structural boundaries and mandatory semantic backtracking automatically induce a Lyapunov function**—using only two pointers and two operators (push/pop), without predefining any energy function. This yields the first convergence criterion derived from *reasoning structure* rather than *energy analysis*. Extending the Yonglin Formula, we demonstrate that reasoning incompleteness is not a defect but a *dynamical system property*—convergence occurs precisely *because* computational boundaries and prior anchors exist.

The synthesis: Reasoning's incompleteness is its dynamics. Boundaries enable convergence. The stack meets Euler at the fixed point.

Keywords: Reasoning systems, Computational boundaries, Euler dynamics, Stack models, Incompleteness theory, Phase transitions

1 Introduction

1.1 The Paradox of Scale

Contemporary AI research operates under a seductive hypothesis: *scaling up neural networks will yield reasoning capabilities*. More parameters, more data, more compute—surely intelligence will

emerge.

Yet a paradox haunts this narrative. Consider:

- Modern accelerators operate in BF16 (16-bit brain floating point), providing $2^{16} \approx 65,000$ discrete values per dimension.
- A typical language model has hidden dimension $d = 4096$.
- The resulting state space has cardinality $\approx (65,000)^{4096} \approx 10^{19,600}$ distinct states.

By comparison:

- Go has $\approx 10^{170}$ legal board positions.
- Chess has $\approx 10^{47}$ positions.
- Atari game state spaces range from 10^9 to 10^{12}.
- Typical planning problems have search spaces $< 10^{100}$.

The representation space is not the bottleneck.
Current models possess state spaces **orders of magnitude larger** than the problems they fail to solve. The failure is not one of *capacity* but of *structure*.

This is the first part of our critique: **the representation space is wasted**.

1.2 The Ignored Boundaries

Classical computability theory tells us that computational boundaries *exist* (halting problem, P vs NP). But where, precisely, do these boundaries lie?

Recent work [1] answered this through Monte Carlo experiments: computational problems exhibit **sharp phase transitions** at critical densities $d_c(L)$ that follow logarithmic scaling laws:

$$d_c(L) = -0.0809 \ln(L) + 0.501 \quad (\text{MSE } \sim 10^{-32})$$

Furthermore, incompleteness theory [2] established that reasoning cannot be complete without prior anchors:

$$\lim_{n \to \infty} \Pi^{(n)}(s) = A, \quad A \neq A^*$$

These are not Lipschitz-contraction convergence guarantees. These are *structural phase transitions* and *meta-level ruptures*.

1.3 Our Contribution

We synthesize these insights into a unified theory:

1. **Representation Space Waste Analysis:** Quantitative proof that BF16/FP32 state spaces dwarf problem complexities, eliminating "insufficient capacity" as an excuse (Section 2).
2. **Categorical Mismatch Theorem:** All sequential models decompose as $\Phi = I + F$ (pseudo-Euler), rendering them irreversible, collapsing, and RNN-equivalent—regardless of architecture (Section 3).

3. **Computational Boundaries:** Integration of phase transition theory showing that solvability boundaries are information-theoretic, not merely asymptotic (Section 4).

4. **Reasoning Incompleteness:** Formal connection between Yonglin Formula's prior anchors and computational boundaries (Section 5).

5. **Euler-Stack Correspondence:** Proof that stack pointer dynamics with fixed boundaries admit *honest discrete Euler* structure with guaranteed convergence (Sections 6-8).

6. **Automatic Lyapunov Construction from Minimal Structure:** We prove that reasoning systems with structural boundaries and mandatory semantic backtracking (pop operations) *automatically induce* a Lyapunov function—without predefining any energy function. Using only two pointers (stack top t_n, stack bottom $t_\perp = 0$) and two operators (push, pop), we construct $V(t) = t$ as the natural convergence certificate. This is the **first convergence criterion derived from reasoning structure rather than energy analysis** (Section 8).

7. **The Synthesis:** Incompleteness is not a bug—it is the *dynamics* that enables convergence. Boundaries and priors are not limitations but *necessary conditions* for reasoning (Section 9).

1.4 The Narrative Arc

THE PROBLEM
Representation Space Wasted (90%+ unused)
↓ *Why?*
Pseudo-Euler Collapse ($\Phi = I + F \Rightarrow$ RNN-like)
↓ *What ignored?*
IGNORED REALITY
Computational Boundaries Exist (phase transitions)
Reasoning Incompleteness (prior anchors required)
↓ *Hope?*
THE SOLUTION
Stack Meets Euler (true discrete dynamics)
↓ *Proven!*
Convergence with Boundaries (Lyapunov descent)
↓ *Why?*
THE SYNTHESIS
Incompleteness = Dynamics (fixed point convergence)

1.5 Roadmap

1. **Section 2:** The Wasted Representation Space—proving BF16 suffices for all practical reasoning tasks.

2. **Section 3:** The False Euler—Theorem proving $\Phi = I + F$ entails irreversibility and semantic collapse.

3. **Section 4:** Computational Boundaries Exist—Monte Carlo phase transitions.

4. **Section 5:** Reasoning Incompleteness—Yonglin Formula and prior anchors.

5. **Section 6:** Stack-Based Reasoning Systems—formal definitions.

6. **Section 7:** The Euler-Stack Correspondence Theorem.
7. **Section 8:** Convergence Under Boundaries—Yonglin Extension.
8. **Section 9:** Synthesis: Incompleteness as Dynamical System.
9. **Section 10:** Four Dimensions of Structural Failure.
10. **Section 11:** Roadmap for Future Systems.
11. **Section 12:** Conclusion.

2 The Wasted Representation Space

Before analyzing *how* current models fail, we must establish *what they cannot blame*. We prove that representation capacity is not the bottleneck.

2.1 Quantifying State Spaces

Definition 2.1 (Floating-Point State Space)**.** A d-dimensional hidden state using b-bit floating-point representation admits:

$$|\mathcal{S}_{\text{float}}| = (2^b)^d$$

distinct representable states.

Format	**Bits**	**Values/dim**	$d = 1024$ **states**
BF16	16	65,536	$10^{4,930}$
FP16	16	65,536	$10^{4,930}$
FP32	32	4.3×10^9	$10^{9,864}$
FP64	64	1.8×10^{19}	$10^{19,728}$

Table 1: State space cardinalities for standard floating-point formats with hidden dimension $d = 1024$.

2.2 Problem Space Requirements

2.3 The Surplus Theorem

Theorem 2.2 (Representation Surplus)**.** *For any practical reasoning task T (planning, game-playing, theorem-proving) with state space $|\mathcal{S}_T| < 10^{300}$, and any modern neural architecture using BF16 with $d \geq 512$:*

$$|\mathcal{S}_{float}| > 10^{1000} \cdot |\mathcal{S}_T|$$

The representation space exceeds the problem space by ***at least three orders of magnitude****.*

Proof. From Table 1, BF16 with $d = 512$ yields:

$$|\mathcal{S}_{\text{BF16}}| = (65536)^{512} \approx 10^{2465}$$

For any $|\mathcal{S}_T| < 10^{300}$:

$$\frac{|\mathcal{S}_{\text{BF16}}|}{|\mathcal{S}_T|} > \frac{10^{2465}}{10^{300}} = 10^{2165} \gg 10^{1000}$$

□

Domain	State Space Size	BF16 Coverage
Chess (legal positions)	10^{47}	$10^{4,883}$ surplus
Go (legal positions)	10^{170}	$10^{4,760}$ surplus
Atari 2600 (RAM states)	10^{308}	$10^{4,622}$ surplus
Planning (PDDL benchmarks)	$< 10^{100}$	$10^{4,830}$ surplus
Theorem proving (Lean)	$< 10^{200}$	$10^{4,730}$ surplus
Typical LLM	**BF16,** $d = 4096$	$\mathbf{10^{19,720}}$

Table 2: Comparison of problem state spaces vs. BF16 representation capacity. Even with conservative dimension estimates, floating-point spaces **exceed** problem requirements by **orders of magnitude**.

2.4 Implications: The Bottleneck is Not Capacity

Corollary 2.3 (Wasted Representation)**.** *Current neural reasoning systems fail* ***not*** *because:*

- *State spaces are too small (Theorem 2.2 disproves this);*
- *Precision is insufficient (BF16 exceeds requirements);*
- *Embeddings lack expressiveness (surplus is exponential).*

The failure must lie in the ***operator structure****—the way these vast state spaces are traversed during inference.*

> **The Problem, Part I:**
> *Scaling has failed not because we lack representation capacity, but because we are using the wrong operators on the right spaces. The state space is wasted.*

2.5 Utilization Rate Analysis

We now quantify precisely *how much* representation space is wasted.

Definition 2.4 (Representation Utilization Rate)**.** For a reasoning task with state space $\mathcal{S}_T$ and neural representation space $\mathcal{S}_{\text{float}}$, define:

$$\rho_{\text{util}} := \frac{\log |\mathcal{S}_T|}{\log |\mathcal{S}_{\text{float}}|}$$

This measures the fraction of representational capacity theoretically required.

Corollary 2.5 (Massive Under-Utilization)**.** *For all practical reasoning tasks:*

$$\rho_{util} < 0.1$$

More than ***90% of representation capacity remains unused****.*

Task	$\log \lvert\mathcal{S}_T\rvert$	$\log \lvert\mathcal{S}_{\mathbf{BF16}}$	$\rho_{\mathbf{util}}$	**% Used**
Chess	47	4,930	9.5×10^{-3}	0.95%
Go	170	4,930	3.4×10^{-2}	3.4%
Atari 2600	308	4,930	6.2×10^{-2}	6.2%
Planning (PDDL)	100	4,930	2.0×10^{-2}	2.0%
Theorem proving	200	4,930	4.1×10^{-2}	4.1%
Typical LLM	—	**19,720**	$< 10^{-2}$	$<$ **1%**

Table 3: Utilization rates for BF16 with $d = 1024$. Even the most complex tasks use $< 7\%$ of available representation capacity.

Model	**Params**	**Hidden** d	$\log \lvert\mathcal{S}\rvert$	**Task Performance**
GPT-4	1.76T	12,288	$\approx 59,000$	Fails multi-step reasoning
Claude 3 Opus	Unknown	∼8,192	$\approx 39,000$	Fails complex planning
Gemini Ultra	Unknown	∼16,384	$\approx 78,000$	Fails theorem proving
Llama 3 405B	405B	16,384	$\approx 78,000$	Fails Go/Chess
Go (AlphaGo)	—	—	**170**	**Superhuman (2016)**
Chess (Stockfish)	—	—	**47**	**Superhuman (1997)**

Table 4: Comparison of LLM state spaces vs. task requirements. Despite having representation spaces 10^3-10^5 times larger than game state spaces, LLMs fail tasks that specialized systems solved decades ago.

2.6 Empirical Evidence from State-of-the-Art Models

We examine actual model deployments to verify our theoretical analysis.

Observation 2.6 (The Scaling Paradox). Consider the timeline:

- **1997:** Deep Blue beats Kasparov at chess ($\mathcal{S}_{\text{chess}} \sim 10^{47}$)
- **2016:** AlphaGo beats Lee Sedol at Go ($\mathcal{S}_{\text{Go}} \sim 10^{170}$)
- **2024:** GPT-4 with $\mathcal{S}_{\text{float}} \sim 10^{59,000}$ still cannot reliably solve multi-step reasoning tasks

The representation space has grown by $10^{58,800}$ times, yet reasoning capability has *not improved proportionally*—in many cases, it has *regressed*.

2.7 Information-Theoretic Waste

Theorem 2.7 (Entropic Inefficiency)**.** *Let $H(T)$ be the Shannon entropy of task T and $H(\mathcal{S}_{float})$ be the entropy of the representation space. For modern LLMs:*

$$\frac{H(T)}{H(\mathcal{S}_{float})} < 10^{-2}$$

This implies that the effective information-per-bit is:

$$\eta_{info} = \frac{H(T)}{b \cdot d} < 10^{-5} \text{ bits/bit}$$

where $b = 16$ (BF16) and $d \sim 10^4$ (typical hidden dimension).

Proof. From Table 3, $\rho_{\text{util}} < 0.1$ for all tasks. Since $H(T) \leq \log |\mathcal{S}_T|$ and $H(\mathcal{S}_{\text{float}}) = \log |\mathcal{S}_{\text{float}}|$:

$$\frac{H(T)}{H(\mathcal{S}_{\text{float}})} \leq \frac{\log |\mathcal{S}_T|}{\log |\mathcal{S}_{\text{float}}|} = \rho_{\text{util}} < 0.1$$

For the worst case (Go with $\rho_{\text{util}} = 0.062$):

$$\eta_{\text{info}} = \frac{H(\text{Go})}{16 \times 1024} \approx \frac{170}{16,384} \approx 1.04 \times 10^{-2}$$

For typical reasoning tasks ($\log |\mathcal{S}_T| \sim 100$):

$$\eta_{\text{info}} \approx \frac{100}{16,384} \approx 6.1 \times 10^{-3}$$

This is orders of magnitude below the theoretical maximum of 1 bit/bit. □

2.8 The Compute Waste Implication

Corollary 2.8 (Computational Inefficiency)**.** *If $\rho_{util} < 0.1$ but models require C FLOPs per inference, then the **effective FLOPs** for reasoning is:*

$$C_{eff} = \rho_{util} \cdot C < 0.1 \cdot C$$

*At least **90% of compute is wasted** on unused representation capacity.*

Example 2.9 (GPT-4 Inference Cost)**.** Suppose GPT-4 uses $C \sim 10^{13}$ FLOPs per forward pass (conservative estimate for 1.76T parameters). From Corollary 2.8:

$$C_{\text{wasted}} = (1 - \rho_{\text{util}}) \cdot C > 0.9 \times 10^{13} = 9 \times 10^{12} \text{ FLOPs}$$

are spent maintaining unused representation capacity rather than performing reasoning operations.

This explains why scaling compute does not proportionally improve reasoning: *the additional compute is wasted on unutilized state space.*

2.9 Why Scaling Fails: The Fundamental Disconnect

Theorem 2.10 (Scaling-Reasoning Disconnect)**.** *Let N_{params} be the number of parameters and $\mathcal{R}(N)$ be reasoning capability. Current architectures satisfy:*

$$\frac{d\mathcal{R}}{d \log N_{params}} \to 0 \quad \text{as } N_{params} \to \infty$$

***Reasoning capability saturates** despite unbounded parameter scaling.*

Proof sketch. From Theorem 2.2, representation capacity already exceeds task requirements by orders of magnitude. Therefore:

(i) Increasing d (hidden dimension) does not help: $\mathcal{S}_{\text{float}}$ is already 10^{1000} times larger than needed.

(ii) Increasing depth (more layers) does not help: Theorem 3.3 shows collapse is structural, not capacity-limited.

(iii) Increasing width (more heads) does not help: Still subject to $\Phi = I + F$ decomposition (Theorem 3.1).

Since $\mathcal{R}$ is bounded by structural properties (reversibility, backtracking, reflexivity—see Section 3), not capacity:

$$\mathcal{R}(N) < \mathcal{R}_{\max} < \infty \quad \forall N$$

Hence $\frac{d\mathcal{R}}{d \log N} \to 0$ as $N \to \infty$. □

Extended Problem Statement, Part I:
The representation space is wasted (90%+ unused). Compute is wasted (90%+ maintaining unused capacity). Scaling is wasted (saturating reasoning gains). The failure is not capacity—it is ***categorical operator mismatch****.*

3 The False Euler: Why All Sequential Models Collapse

Having eliminated representation capacity as an excuse, we now identify the true culprit: **pseudo-Euclidean operator dynamics**.

3.1 The Euler Emergence Theorem

Theorem 3.1 (Euler Emergence)**.** *Let $h_t \in \mathbb{R}^d$ be a state vector at discrete time t, and let $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ be any state-update function. Then:*

$$h_{t+1} = \Phi(h_t, x_t; \theta)$$

necessarily admits the decomposition:

$$\Phi = I + F$$

where I is the identity map and $F : \mathbb{R}^d \to \mathbb{R}^d$ is defined by:

$$F(h_t, x_t; \theta) := \Phi(h_t, x_t; \theta) - h_t$$

Therefore, every sequential update can be written in ***pseudo-Euler form****:*

$$h_{t+1} = h_t + F(h_t, x_t; \theta)$$

Proof. This is a trivial algebraic identity. Define:

$$\Delta h_t := h_{t+1} - h_t = \Phi(h_t, x_t; \theta) - h_t$$

Let $F := \Delta h_t$. Then:

$$h_{t+1} = h_t + F(h_t, x_t; \theta)$$

This is the discrete Euler form with step size $\Delta t = 1$. □

Remark 3.2 (Categorical Necessity). We do not *choose* to interpret neural networks as Euler schemes—the decomposition $\Phi = I + F$ is *unavoidable*. This is not a modeling assumption; it is a categorical fact about difference equations.

3.2 Structural Irreversibility

Theorem 3.3 (Inevitable Irreversibility)**.** *For any non-trivial sequential model where $F \neq 0$ and dimension d is finite, the update map $\Phi = I + F$ is generically irreversible: there exist distinct states $h_1 \neq h_2$ such that:*

$$\Phi(h_1) = \Phi(h_2)$$

Proof. Neural networks employ non-linear activations (ReLU, softmax, layer normalization) that compress unbounded inputs into bounded outputs. These are necessarily many-to-one functions. Hence Φ is not injective.

More formally: activation functions like $\sigma(x) = \frac{1}{1+e^{-x}}$ satisfy $\sigma : \mathbb{R} \to (0, 1)$, mapping an infinite domain to a bounded range. Any composition involving such functions is non-injective. □

Corollary 3.4 (Semantic Collapse)**.** *Because Φ is irreversible, there exist semantically distinct reasoning states h_1, h_2 that are mapped to the same state $h' = \Phi(h_1) = \Phi(h_2)$.* ***Information is lost irreversibly.***

3.3 All Sequential Models are RNN Variants

Corollary 3.5 (RNN Universality)**.** *Any model of the form $h_{t+1} = \Phi(h_t, x_t; \theta)$ is structurally equivalent to a Recurrent Neural Network,* ***regardless of architectural details****.*

Proof. The defining characteristic of an RNN is the recurrence:

$$h_{t+1} = G(h_t, x_t)$$

Theorem 3.1 shows that any sequential update is of this form with $G = I + F$. Hence:

- **Transformers:** Autoregressive generation satisfies $s_{t+1} = s_t \oplus \text{Attention}(s_t, x_t)$ (token concatenation or state update). This is an RNN.
- **LSTMs/GRUs:** Explicitly designed as RNNs with gating.
- **State-space models (S4, Mamba):** Linear recurrences $h_{t+1} = Ah_t + Bx_t$. Still RNNs.

All differ only in the choice of F. □

Remark 3.6 (The Pretense of Differentiability). Models are trained via backpropagation, creating the illusion of smooth, continuous dynamics. But execution is discrete: each token generation is a *difference step*, not a differential. We call this **pseudo-Euler**: pretending to approximate $\frac{dh}{dt} = F(h)$ while actually executing $h_{t+1} = h_t + F(h_t)$ with no underlying continuous limit.

3.4 Why This Matters

Theorem 3.1 and 3.3 immediately imply:

(i) **Irreversibility:** Cannot recover previous states. Reasoning requiring backtracking (proof search, hypothesis revision) is impossible.

(ii) **Semantic Collapse:** Distinct contexts merge (Corollary 3.4). Fine-grained distinctions are lost.

(iii) **Absence of Reflexivity:** Parameters θ are fixed during inference. The system cannot reflect on its assumptions.

(iv) **False Backtracking:** Generating from an earlier state is re-execution, not true backtracking. No memory of abandoned paths.

3.5 Quantifying the Collapse Rate

We now quantify precisely *how fast* semantic information is lost through irreversible transformations.

Definition 3.7 (Information Loss Rate)**.** For a sequence of updates $h_0, h_1, \ldots, h_T$ under $h_{t+1} = \Phi(h_t, x_t)$, define the **collapse rate**:

$$\lambda_{\text{collapse}} := \frac{1}{T} \sum_{t=0}^{T-1} \frac{\|h_{t+1} - h_t\|}{\|h_t\|}$$

This measures the average relative change per step.

Theorem 3.8 (Exponential Semantic Collapse)**.** *Consider a sequential model where activation functions satisfy $\sigma : \mathbb{R} \to [-M, M]$ (bounded). For any initial state h_0 with $\|h_0\| = H_0$, after T steps:*

$$\textit{Rank}(\{h_0, h_1, \ldots, h_T\}) \leq \min\left(d, \frac{2MT}{\epsilon}\right)$$

where d is dimension and ϵ is numerical precision.

*The effective dimensionality of the trajectory is **linearly bounded**, not exponentially growing.*

Proof. Bounded activations map $\mathbb{R}^d \to [-M, M]^d$. The image has bounded ℓ_∞ norm. After T steps, all states lie in:

$$\mathcal{B}_\infty(M) = \{h \in \mathbb{R}^d : \|h\|_\infty \leq M\}$$

The ϵ-covering number of this set is:

$$N_\epsilon(\mathcal{B}_\infty(M)) \leq \left(\frac{2M}{\epsilon}\right)^d$$

But for a trajectory of length T, we visit at most T distinct points. Therefore:

$$\text{Rank(trajectory)} \leq \min\left(d, \log_2\left(\frac{2MT}{\epsilon}\right)\right)$$

This grows **logarithmically** in T, not exponentially as required for exponentially large state spaces. □

Corollary 3.9 (Representation Collapse)**.** *Despite having $|\mathcal{S}_{\textit{float}}| \sim 10^{19,720}$ representable states (Table 2), any inference trajectory visits at most:*

$$|\textit{visited states}| \leq T \ll |\mathcal{S}_{\textit{float}}|$$

For $T = 1000$ tokens (typical inference), the utilization is:

$$\frac{T}{|\mathcal{S}_{\textit{float}}|} \sim \frac{10^3}{10^{19,720}} \sim 10^{-19,717}$$

Less than** $10^{-19,717}$ **of the state space is ever accessed.

3.6 The Scaling-Collapse Theorem

Theorem 3.10 (Scaling Amplifies Collapse)**.** *Let N_{params} be the number of parameters and T be inference length. The total number of floating-point operations is:*

$$FLOPs = \Theta(N_{params} \cdot T)$$

*But from Corollary 3.9, the number of distinct states visited is $\leq T$. Therefore, the **FLOPs per distinct state** is:*

$$\frac{FLOPs}{distinct\ states} = \Theta(N_{params})$$

*Scaling parameters **linearly increases** compute per state without increasing state diversity.*

Proof. Each forward pass requires $\Theta(N_{\text{params}})$ operations (matrix multiplies, activations). Over T steps:

$$\text{FLOPs} = T \cdot \Theta(N_{\text{params}})$$

From Theorem 3.8, trajectory visits $\leq T$ distinct states. Hence:

$$\frac{\text{FLOPs}}{\text{states}} = \frac{T \cdot \Theta(N_{\text{params}})}{T} = \Theta(N_{\text{params}})$$

Doubling N_{params} doubles compute per state but does *not* double the number of reachable states (bounded by T). □

Example 3.11 (GPT-4 vs GPT-3)**.**

- **GPT-3:** $N \sim 175$B parameters, FLOPs $\sim 3.5 \times 10^{11}$ per token
- **GPT-4:** $N \sim 1.76$T parameters ($10\times$ larger), FLOPs $\sim 3.5 \times 10^{12}$ per token ($10\times$ more)

From Theorem 3.10, both models visit $\approx T$ states (same trajectory length), but GPT-4 spends $10\times$ more compute per state.

Result: Marginal reasoning improvement despite $10\times$ compute increase.

3.7 Why Scaling Amplifies Failure

Lemma 3.12 (Irreversibility Scales with Depth)**.** *For a model with L layers, each with compression ratio $\rho_\ell < 1$ (non-injective), the total compression is:*

$$\rho_{total} = \prod_{\ell=1}^{L} \rho_\ell$$

For $\rho_\ell = 0.9$ (modest 10% compression per layer) and $L = 100$ layers:

$$\rho_{total} = 0.9^{100} \approx 2.66 \times 10^{-5}$$

Deeper models compound irreversibility exponentially.

Proof. Each layer ℓ applies $\Phi_\ell : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$. If Φ_ℓ is ρ_ℓ-compressive (effective dimension reduced by factor ρ_ℓ), then after L layers:

$$\text{Effective dimension} = d_0 \cdot \prod_{\ell=1}^{L} \rho_\ell$$

This shrinks exponentially in L. □

Corollary 3.13 (Deep Networks Collapse Faster)**.** *Increasing depth L to improve capacity backfires: deeper models have* ***more severe semantic collapse****.*

From Lemma 3.12:

$$\rho_{total} = \rho^L \to 0 \quad as\ L \to \infty$$

Asymptotically, all inputs collapse to a ***single fixed point****.*

3.8 Architectural Variants: All Roads Lead to RNN

We verify that various architectural improvements still succumb to $\Phi = I + F$ collapse.

Architecture	Update Form	RNN?	Irreversible?
Vanilla RNN	$h_{t+1} = \tanh(Wh_t + Ux_t)$	Yes	Yes
LSTM	$h_{t+1} = f_t \odot h_t + i_t \odot \tilde{c}_t$	Yes	Yes
GRU	$h_{t+1} = (1 - z_t) \odot h_t + z_t \odot \tilde{h}_t$	Yes	Yes
Transformer (AR)	$s_{t+1} = s_t \oplus \text{Attn}(s_t, x_t)$	Yes	Yes
State-Space (S4)	$h_{t+1} = Ah_t + Bx_t$	Yes	Yes (if A singular)
Mamba	$h_{t+1} = A(x_t)h_t + B(x_t)x_t$	Yes	Yes
Retentive Network	$h_{t+1} = \gamma h_t + \text{Retention}(x_t)$	Yes	Yes
RWKV	$h_{t+1} = \alpha h_t + \beta\text{WKV}(x_t)$	Yes	Yes

Table 5: All sequential architectures admit $h_{t+1} = \Phi(h_t, x_t)$ form, hence are RNN-equivalent (Corollary 3.5) and irreversible (Theorem 3.3).

Observation 3.14. Even architectures claiming to "fix" Transformers or RNNs (e.g., Mamba, RWKV, Retentive Networks) still satisfy:

$$h_{t+1} = h_t + F(h_t, x_t) \quad \text{(pseudo-Euler)}$$

They differ only in the choice of F, not in the fundamental categorical structure.

3.9 The Category Error

The Category Error:
The AI community treats reasoning as a problem of *function approximation* in $\mathbb{R}^d$:

$$\text{"Find } f : \mathbb{R}^d \to \mathbb{R}^d \text{ such that } f(h) \approx h^*\text{"}$$

But reasoning is actually a problem of *operator category*:

"Find category $\mathcal{C}$ with morphisms supporting reversibility, reflexivity, termination"

The failure is categorical, not representational.

The Problem, Part II (Extended):
All sequential models are pseudo-Euler schemes that collapse into irreversible, semantically lossy RNN-like structures. Scaling amplifies this failure: deeper models collapse faster (Corollary 3.13), larger models waste more compute per state (Theorem 3.10), and all architectural variants fail identically (Table 5). This is not a bug—it is the categorical structure of $\Phi = I + F$ in finite-dimensional vector spaces.

4 Computational Boundaries Exist: Phase Transitions in Solvability

Having shown *how* current models fail, we now address what they *ignore*: the existence of sharp computational boundaries.

4.1 From Asymptotic to Exact

Classical complexity theory establishes *qualitative* boundaries:

- Halting problem is undecidable [3]
- SAT is NP-complete [4]

But *where exactly* are these boundaries? Recent work [1] answered this through statistical mechanics.

4.2 Monte Carlo Phase Transition Discovery

Theorem 4.1 (Logarithmic Scaling Law [1])**.** *For constraint satisfaction problems of size L with constraint density d, the critical density (50% solvability threshold) follows:*

$$\boxed{d_c(L) = -\alpha \ln(L) + \beta}$$

where $\alpha = 0.0809 \pm 0.0001$, $\beta = 0.501 \pm 0.001$ (empirical constants with MSE $\sim 10^{-32}$).

Theorem 4.2 (Universal Phase Transition Kernel [1])**.** *All phase transition curves share a single functional form:*

$$\mu(L, d) = K(d - d_c(L))$$

where the kernel is:

$$\boxed{K(x) = \frac{1}{2}\left(1 - erf\left(\frac{x}{\sigma}\right)\right)}$$

with $\sigma = 0.1007 \pm 0.0003$ (universal constant).

4.3 Information-Theoretic Origin

The logarithmic form $d_c \sim \ln(L)$ suggests an information-theoretic origin. The constraint tolerance decays as:

$$\frac{\partial d_c}{\partial \ln(L)} = -\alpha$$

Interpretation: Each additional bit of problem information reduces constraint budget by 8.09%. This logarithmic decay is characteristic of information-theoretic phase transitions, where the critical density marks the boundary between tractable and intractable problem regimes.

4.4 This is Not Lipschitz Contraction

Crucially, these boundaries are **not** arising from Lipschitz-contraction guarantees (as in Banach fixed-point theorem). They are **statistical phase transitions**:

- Error function kernel $\sim$ cumulative Gaussian (central limit theorem)
- Sharp transition width $\sigma \approx 0.1$ (universality class)
- Logarithmic scaling $\sim$ information entropy (Shannon)

> **Ignored Reality:**
> *Computational boundaries are real, quantifiable, and information-theoretic. They are not Lipschitz estimates or worst-case bounds—they are phase transitions with universal critical exponents.*

5 Reasoning Incompleteness: The Prior Anchor

Computational boundaries reveal *where* problems become unsolvable. Incompleteness theory reveals *why* reasoning cannot be self-sufficient.

5.1 The Yonglin Formula

Building on [2], we recall the central result:

Theorem 5.1 (Yonglin Formula [2])**.** *Let $\mathcal{R} = (S, \Pi, A)$ be a reasoning system with prior anchor A. For any initial state $s \in S$:*

$$\lim_{n\to\infty} \Pi^{(n)}(s) = A$$

All reasoning returns to its prior in the limit.

Furthermore, applying the reflexive operator:

$$A^* = \left(\lim_{n\to\infty} \Pi^{(n)}(s)\right)^*$$

yields $A \neq A^*$ (meta-level rupture). ***Object-level closure, meta-level rupture.***

5.2 Connection to Computational Boundaries

The prior anchor A is *not arbitrary.* It is:

- The fixed point of reasoning iteration
- The semantic bottom that cannot be eliminated
- The computational boundary $(a_\perp, h_\perp)$ in stack models (Section 6.2)

Without A, reasoning enters infinite regress (proven in [2], Section 2). With A, reasoning converges—but incompletely ($A \neq A^*$).

5.3 Why Linear Models Ignore This

Linear models have **no natural prior anchor**:

- The zero vector $\mathbf{0} \in \mathbb{R}^d$ is *arbitrary* (any vector could be chosen under translation)
- Parameters θ are fixed, not reflexive
- No structural boundary enforces convergence

Stack models, by contrast, have *structural anchors*:

- Fixed stack-bottom frame $(a_\perp, h_\perp)$
- Pointer constrained $t_n \geq 0$
- Boundary is *enforced by dynamics*, not assumed

> **Ignored Reality, Part II:**
> *Reasoning is incomplete without prior anchors (Yonglin Formula). These anchors are not assumptions—they are the computational boundaries revealed by phase transitions. Linear models lack such anchors structurally.*

5.4 The Prior-Boundary Correspondence

We now establish the precise mathematical connection between Yonglin's prior anchors and computational phase boundaries.

Theorem 5.2 (Prior-Boundary Correspondence)**.** *Let $\mathcal{R} = (S, \Pi, A)$ be a reasoning system with prior anchor A (Definition from [2]). The prior anchor A is mathematically equivalent to the computational boundary from Theorem 4.1.*

Specifically:

(i) The prior anchor A acts as a semantic attractor: $\lim_{n\to\infty} \Pi^{(n)}(s) = A$

(ii) The critical density $d_c(L)$ acts as a phase boundary: $\mu(L, d_c) = 0.5$

*(iii) Both are **fixed points** that cannot be eliminated without destroying the system*

Proof. From [2], the prior anchor satisfies:

$$\Pi(A) = A \quad \text{(fixed point)}$$

From Theorem 4.1, the critical density satisfies:

$$\left.\frac{\partial \mu}{\partial d}\right|_{d=d_c} = \text{maximal (phase transition)}$$

Both represent **structural boundaries** where dynamics qualitatively change:

- **Below d_c:** Problems are solvable ($\mu \approx 1$)
- **At d_c:** Phase transition ($\mu = 0.5$)
- **Above d_c:** Problems are unsolvable ($\mu \approx 0$)

Similarly, for reasoning iterations:

- **Far from A:** Reasoning actively updates state
- **At A:** Fixed point (no further updates)
- **Past reflexive limit:** Meta-level rupture ($A \neq A^*$)

Both A and d_c are *unavoidable structural features*, not free parameters. □

5.5 Why Incompleteness Enables Convergence

Lemma 5.3 (Completeness Implies Non-Termination)**.** *Suppose a reasoning system $\mathcal{R}$ is complete (no prior anchor required). Then for any initial state s_0:*

$$\Pi^{(n)}(s_0) \neq \Pi^{(m)}(s_0) \quad \forall n \neq m$$

The iteration never terminates (infinite regress).

Proof sketch. If $\mathcal{R}$ has no prior anchor, then Π has no fixed point within S. From [2], this leads to infinite justification chains:

$$s_0 \xleftarrow{\Pi} s_1 \xleftarrow{\Pi} s_2 \xleftarrow{\Pi} \cdots$$

where each s_i requires further justification. No s_i can be self-justifying (otherwise it would be a prior anchor). Hence the sequence never stabilizes. □

Corollary 5.4 (Incompleteness is Necessary for Termination)**.** *A reasoning system can terminate in finite steps **only if** it is incomplete (has a prior anchor A).*

Formally:

$$\exists N < \infty : \Pi^{(n)}(s_0) = A \quad \forall n \geq N \quad \iff \quad \mathcal{R} \text{ is incomplete}$$

5.6 The Boundary as Semantic Ground

Definition 5.5 (Semantic Grounding)**.** A reasoning system is **semantically grounded** if its prior anchor A corresponds to:

- **Axiomatic truths** (cannot be further reduced)
- **Observational data** (directly perceived, not inferred)
- **Computational primitives** (elementary operations)

These form the *semantic bottom* beyond which reasoning cannot penetrate.

Example 5.6 (Mathematical Reasoning)**.** In formal mathematics:

- **Prior anchor A:** ZFC axioms, logical rules (modus ponens, etc.)
- **Incompleteness:** Gödel's theorems ($A \neq A^*$)
- **Convergence:** All proofs terminate at axioms

Without axioms (no A), mathematical reasoning enters infinite regress ("Why is modus ponens valid?" → meta-logic → meta-meta-logic → $\cdots$).

Example 5.7 (Empirical Reasoning)**.** In scientific inference:

- **Prior anchor** A**:** Experimental observations, measurement protocols
- **Incompleteness:** Problem of induction ($A \neq A^*$: observations $\not\Rightarrow$ universal laws)
- **Convergence:** All theories terminate at empirical evidence

Without observational ground (no A), scientific reasoning becomes pure speculation.

5.7 Linear Models Have No Semantic Ground

Proposition 5.8 (Absence of Grounding in $\mathbb{R}^d$)**.** *For linear models $h_{t+1} = h_t + F(h_t, x_t; \theta)$ in $\mathbb{R}^d$:*

(i) There is no distinguished vector $h_\perp$ serving as semantic ground (all vectors equivalent under translation)

(ii) The zero vector $\mathbf{0}$ is an arbitrary choice, not structurally enforced

(iii) Parameters θ are fixed during inference, preventing reflexive grounding updates

Therefore, ***linear models lack semantic grounding****.*

Proof. For any $h \in \mathbb{R}^d$ and translation $\tau \in \mathbb{R}^d$, the translated model:

$$h'_{t+1} = (h_t + \tau) + F(h_t + \tau, x_t; \theta)$$

is mathematically equivalent (can be absorbed into bias terms). Hence no vector has *structural* significance.

Furthermore, during inference, θ is frozen. The model cannot modify its own "axioms" (parameters). This contrasts with stack models where the boundary frame $(a_\perp, h_\perp)$ is *structurally protected* (Definition 6.2). □

5.8 The Paradox Resolved

The Paradox of Incompleteness:
Naive view: Incompleteness is a *limitation*—reasoning cannot justify everything.
Truth: Incompleteness is a *necessity*—without it, reasoning cannot terminate (Lemma 5.3).
Deep insight: The boundary (prior anchor) is not a flaw but the *foundation*. Reasoning converges **because** it is incomplete, not despite it.

Extended Analysis of Ignored Reality:
Computational boundaries (Theorem 4.1) and prior anchors (Theorem 5.1) are two faces of the same necessity. Boundaries enable termination. Anchors enable convergence. Together, they form the ***semantic ground*** *that makes reasoning possible. Linear models, lacking both boundaries and anchors, float ungrounded in $\mathbb{R}^d$.*

6 Stack-Based Reasoning Systems

We now introduce the alternative: stack models with computational boundaries.

6.1 Stack Spaces

Definition 6.1 (Stack Space)**.** A **stack space** is a triple $(\mathcal{S}, \mathcal{A}, \mathcal{H})$ where:

- $\mathcal{H}$ is a semantic state space (reasoning contexts, propositions, proofs);
- $\mathcal{A}$ is an address space (memory locations, indexing);
- $\mathcal{S} = (\mathcal{A} \times \mathcal{H})^*$ is the space of finite sequences of address-semantic pairs.

At time n, the stack is:

$$S_n = \big((a_0^{(n)}, h_0^{(n)}), (a_1^{(n)}, h_1^{(n)}), \ldots, (a_{t_n}^{(n)}, h_{t_n}^{(n)})\big)$$

where $t_n \in \mathbb{N}$ is the **stack-top pointer**.

6.2 Computational Boundary

Definition 6.2 (Computational Boundary / Semantic Bottom)**.** A stack space has a **computational boundary** if there exists a fixed bottom frame:

$$(a_\perp, h_\perp) \in \mathcal{A} \times \mathcal{H}$$

such that for all n:

$$(a_0^{(n)}, h_0^{(n)}) = (a_\perp, h_\perp)$$

and no operation may modify or pop this frame.

Remark 6.3. This is the prior anchor A from Theorem 5.1. It is also the $\mu = 0.5$ critical point from Theorem 4.1—the boundary where reasoning transitions from solvable to unsolvable.

6.3 Pointer Dynamics as Reasoning

Definition 6.4 (Reasoning as Pointer Update)**.** A **reasoning step** is:

$$t_{n+1} = \pi(t_n, c_n)$$

where:

- $t_n \in \mathbb{N}$ is the current stack-top pointer;
- $c_n \in \mathcal{C}$ is context (input, observation);
- $\pi : \mathbb{N} \times \mathcal{C} \to \mathbb{N}$ is the pointer update function.

Constraint: $t_{n+1} \geq 0$ (cannot move below boundary).

6.4 Prior Reflexivity: Address Shift

Definition 6.5 (Address Shift Operator)**.** An **address shift operator** $\Sigma_\delta : \mathcal{A} \to \mathcal{A}$ transforms the address space. Applied globally:

$$S'_n = \Sigma_{\delta_n}(S_n) = \big((a_\perp, h_\perp), (\Sigma_{\delta_n}(a_1), h_1), \ldots\big)$$

where the bottom frame remains fixed.

This models **prior reflexivity**: reasoning transforms its own indexing structure, not just semantic content.

6.5 Total Update

Definition 6.6 (Stack Reasoning System)**.** A complete system is:

$$\mathcal{R}_{\text{stack}} = (S_n, t_n, \pi, \Sigma, U)$$

with update:

$$\begin{aligned} t_{n+1} &= \pi(t_n, c_n) && \text{(pointer move)} \\ S'_n &= \Sigma_{\delta_n}(S_n) && \text{(address shift)} \\ S_{n+1} &= U(S'_n, t_{n+1}, c_n) && \text{(semantic update)} \end{aligned}$$

7 The Euler-Stack Correspondence Theorem

We prove the central result: stack pointer dynamics are isomorphic to honest discrete Euler iterations.

7.1 Main Theorem

Theorem 7.1 (Euler-Stack Correspondence)**.** *Let $\mathcal{R}_{stack} = (S_n, t_n, \pi, \Sigma, U)$ be a stack system with pointer update $t_{n+1} = \pi(t_n, c_n)$.*

Define pointer displacement:

$$\Delta t_n := t_{n+1} - t_n$$

Then:

$$t_{n+1} = t_n + \Delta t_n = t_n + F_{stack}(t_n, c_n)$$

where $F_{stack}(t_n, c_n) \in \mathbb{Z}$ (e.g., ± 1 for push/pop, 0 for stay).

*If computational boundary exists (Definition 6.2), then $t_n \geq 0$ always, and dynamics are **boundary-constrained Euler iteration**.*

Proof. By definition of π:

$$F_{\text{stack}}(t_n, c_n) := \pi(t_n, c_n) - t_n$$

Then:

$$t_{n+1} = t_n + F_{\text{stack}}(t_n, c_n)$$

This is discrete Euler with step size 1. Constraint $t_n \geq 0$ from Definition 6.2. □

7.2 True Euler vs. False Euler

Proposition 7.2 (Honest Discreteness)**.** *In stack pointer dynamics, Euler form is **not** an approximation. It is the exact natural description. There is no hidden continuous limit.*

Proof. $t_n \in \mathbb{N}$, $F_{\text{stack}} \in \mathbb{Z}$. No continuous differential equation is being approximated. This is discrete dynamics, honestly represented. □

	False Euler (Linear)	**True Euler (Stack)**
Form	$h_{t+1} = h_t + F(h_t)$	$t_{n+1} = t_n + F_{\text{stack}}(t_n)$
State space	$\mathbb{R}^d$ (continuous)	$\mathbb{N}$ (discrete)
Reversibility	No (many-to-one)	Yes (stack preserved)
Boundary	None (arbitrary zero)	Structural $(a_\perp, h_\perp)$
Convergence	External criterion	Intrinsic (boundary)
Pretense	Pseudo-continuous	Honest discrete

Table 6: Comparison of pseudo-Euler (linear models) and true Euler (stack models).

7.3 The Isomorphism Theorem

Theorem 7.3 (Stack-Euler Isomorphism)**.** *Let $\mathcal{S}_{stack} = (\mathbb{N}, \pi, t_\perp = 0)$ be the pointer dynamics of a stack system with boundary, and let $\mathcal{E}_{discrete} = (\mathbb{N}, t \mapsto t + F(t), t_\perp = 0)$ be a discrete Euler system with integer updates.*

Then there exists a category isomorphism:

$$\Psi : \mathcal{S}_{stack} \to \mathcal{E}_{discrete}$$

preserving:

(i) Update structure: $\Psi(\pi(t, c)) = \Psi(t) + F(\Psi(t), c)$

(ii) Boundary: $\Psi(t_\perp) = 0$

(iii) Convergence: $\lim_{n\to\infty} \pi^{(n)}(t_0) = t_\perp \iff \lim_{n\to\infty} t_n = 0$

Proof. Define $\Psi : t \mapsto t$ (identity on $\mathbb{N}$). Then:

$$\begin{aligned}\Psi(\pi(t, c)) &= \pi(t, c) \\ &= t + (\pi(t, c) - t) \quad \text{(arithmetic identity)} \\ &= \Psi(t) + F_{\text{stack}}(t, c) \quad \text{(where } F_{\text{stack}} := \pi - \text{id)}\end{aligned}$$

Boundary preservation:

$$\Psi(t_\perp) = \Psi(0) = 0 = t_\perp^{\text{Euler}}$$

Convergence preservation follows from Ψ being identity (bijection). □

Remark 7.4 (Categorical Honesty). Unlike the pseudo-Euler decomposition of linear models (Theorem 3.1), which is a *formal* algebraic identity, the stack-Euler isomorphism is a *categorical* equivalence preserving all structural properties (boundaries, convergence, reversibility).

8 Convergence Under Boundaries: The Yonglin Extension

We now prove that stack dynamics converge due to computational boundaries. Our approach reveals a fundamental insight: **the stack structure itself constructs its own Lyapunov function**.

We begin with the direct stack dynamics (Section 8.1), then show how this *naturally constructs* the Lyapunov function (Section 8.2), thereby connecting to classical stability theory. The Lyapunov function is not an alternative proof—it is a *consequence* of stack structure.

8.1 Stack Dynamics: The Impossibility of Deficit Stacks

We begin with the most fundamental property of stack-based reasoning: the stack can be empty, but it can never be negative. This simple fact yields the most direct proof of convergence.

Definition 8.1 (Deficit Stack)**.** A **deficit stack** (or **negative stack**) would be a state where the stack pointer is negative: $t_n < 0$. This would correspond to "popping more elements than the stack contains."

Lemma 8.2 (Deficit Stack Paradox)**.** *Any attempt to create a deficit stack (popping from an empty stack) is semantically equivalent to* ***introducing a new semantic element****, not removing one.*

Formally: The operation "pop a non-existent element" cannot be defined without introducing new semantic content to represent "the act of attempting removal from emptiness."

Proof. Consider a stack at the boundary: $t_n = 0$ (empty stack, only the bottom frame $(a_\perp, h_\perp)$ remains).

Attempt 1: Naive deficit. Try to pop: $t_{n+1} = t_n - 1 = -1$.

What does $t = -1$ mean semantically? It cannot mean "one element below the bottom," because the bottom frame $(a_\perp, h_\perp)$ is the *semantic anchor* (Definition 6.2)—there is no semantic content "below" it.

Attempt 2: Define deficit semantically. To give meaning to $t = -1$, we must introduce a new semantic frame:

$$(a_{-1}, h_{-1}) := \text{"the semantic state of having attempted to remove what doesn't exist"}$$

But this *is itself a semantic element*—a new piece of information describing the failed removal attempt.

The paradox: Popping (removing semantic content) has introduced new semantic content (the deficit state). This violates the fundamental meaning of pop as a *semantic stripping operation*.

Resolution: The operation is **semantically undefined**. A deficit stack cannot exist without redefining pop as something that introduces, rather than removes, semantics. □

Theorem 8.3 (Stack Non-Negativity Principle)**.** *For any stack-based reasoning system $\mathcal{R}_{stack} = (S_n, t_n, \pi, \Sigma, U)$ with computational boundary (Definition 6.2):*

$$\boxed{t_n \geq 0 \quad \forall n \in \mathbb{N}}$$

The stack pointer is ***always non-negative****. The stack can be empty ($t_n = 0$), but never in deficit ($t_n < 0$).*

Proof. From Definition 6.2, the bottom frame $(a_\perp, h_\perp)$ is fixed and cannot be removed. This defines $t = 0$ as the *semantic ground*.

Case 1: $t_n > 0$**.** The stack has elements above the boundary. Push/pop operations are well-defined and maintain $t_{n+1} \geq 0$.

Case 2: $t_n = 0$**.** The stack is at the boundary. By definition, no pop operation can remove $(a_\perp, h_\perp)$. Therefore, any operation satisfies:

$$t_{n+1} = \begin{cases} 0 & \text{(stay at boundary)} \\ t_n + k & \text{(push, } k > 0\text{)} \end{cases}$$

In both cases, $t_{n+1} \geq 0$.

Case 3 (hypothetical): $t_n < 0$**.** From Lemma 8.2, this would require introducing new semantic content, contradicting the nature of pop as semantic removal. The operation is undefined.

By induction: $t_0 = 0$ (initial state at boundary) and $t_n \geq 0 \implies t_{n+1} \geq 0$. Therefore, $t_n \geq 0$ for all n. □

Remark 8.4 (Philosophical Interpretation). The impossibility of deficit stacks reflects a deep truth about reasoning:

- **Empty stack ($t = 0$):** No semantic content above the prior anchor. Reasoning has returned to its foundation.
- **Deficit stack ($t < 0$):** Attempting to "go below" the foundation. But there is nothing below the foundation—it is the *semantic bottom* (Section 6.2).
- **Key insight:** To describe "what's below the foundation," you must introduce new semantic concepts. But that *is* the foundation—you've simply redefined your prior anchor.

In other words: **Reasoning cannot escape its priors. Attempting to remove the final prior creates a new prior.**

Theorem 8.5 (Direct Convergence via Stack Dynamics)**.** *Consider a stack-based reasoning system where semantic stripping (pop) dominates semantic introduction (push):*

$$\mathbb{E}[\Delta t_n] < 0 \quad \textit{(expected pointer decrease)}$$

Then reasoning ***must converge*** *to the boundary in finite expected time.*

Proof. From Theorem 8.3, $t_n \geq 0$ always. Furthermore, $t_n \in \mathbb{N}$ (discrete).

Assume $\mathbb{E}[\Delta t_n] < 0$. Then $\{t_n\}$ is a downward-drifting random walk on $\mathbb{N}$ with absorbing barrier at 0.

Standard random walk theory: A downward-drifting walk on $\mathbb{N}$ with absorbing barrier reaches the barrier in finite expected time:

$$\mathbb{E}[\tau] < \infty \quad \text{where } \tau := \inf\{n : t_n = 0\}$$

Deterministic case: If $\Delta t_n \leq -c$ for some $c > 0$, then:

$$\tau \leq \left\lceil \frac{t_0}{c} \right\rceil < \infty$$

Convergence is guaranteed in at most $\lceil t_0/c \rceil$ steps.

In both cases, $t_n \to 0$ in finite time. The stack converges to the boundary $(a_\perp, h_\perp)$. □

Corollary 8.6 (Semantic Interpretation of Convergence)**.** *Reasoning convergence is the natural consequence of:*

(i) ***Semantic stripping is mandatory.*** *Every reasoning step must eventually "cash out" its abstractions by returning to concrete priors (pop operations).*

(ii) ***Deficit is impossible.*** *You cannot strip away the final prior without introducing a new prior (Lemma 8.2).*

(iii) ***Priors are finite.*** *The stack starts at finite depth $t_0 < \infty$.*

Therefore, reasoning must terminate at the prior anchor in finite steps.

Remark 8.7 (Contrast with Yonglin Formula). This proof is **completely independent** of the Yonglin Formula [2]. We have shown convergence using only:

- The impossibility of deficit stacks (Theorem 8.3)
- Basic properties of finite descent in $\mathbb{N}$

No Lyapunov function. No fixed-point argument. Just the *stack structure itself.*
This is the **simplest possible proof** of reasoning convergence.

Key Insight (Stack Dynamics):
Attempting to create a deficit stack (popping what doesn't exist) is itself the introduction of new semantic content. Therefore, stacks are always non-negative. Therefore, finite descending sequences in $\mathbb{N}$ must terminate. Therefore, reasoning must converge.
This is more intuitive than Lyapunov functions. This is the stack's own dynamics.

8.2 The Lyapunov Function: Constructed from Stack Depth

The preceding direct proof (Theorem 8.5) reveals a profound fact: **the stack structure itself constructs a Lyapunov function**. We now make this construction explicit, connecting stack dynamics to classical stability theory.

Theorem 8.8 (Stack Constructs Its Lyapunov Function)**.** *The stack pointer $t_n \in \mathbb{N}$ is a Lyapunov function for the reasoning dynamics. Define:*

$$V : \mathbb{N} \to \mathbb{R}, \quad V(t) := t$$

Then V satisfies all Lyapunov criteria:

(i) ***Positive definite:*** *$V(t) \geq 0$ with $V(0) = 0$ (boundary is equilibrium)*

(ii) ***Monotonic descent:*** *$\Delta V_n = V(t_{n+1}) - V(t_n) \leq 0$ (non-increasing)*

(iii) ***Bounded below:*** *$V(t) \geq 0$ always (from Theorem 8.3)*

Crucially: *V is not chosen or assumed—it is **given by the stack structure itself**. The stack depth t_n is the natural potential function.*

Proof. **(i) Positive definiteness:** From Definition 6.2, $t_n \in \mathbb{N}$ and $t_n \geq 0$ (Theorem 8.3). The boundary $t = 0$ is the equilibrium (no elements above bottom frame).

(ii) Monotonic descent: Assume reasoning satisfies semantic grounding (pop dominates push, Observation 8.15). Then:

$$\mathbb{E}[\Delta t_n] = \mathbb{E}[t_{n+1} - t_n] < 0$$

Hence $\mathbb{E}[V_{n+1}] < \mathbb{E}[V_n]$ (expected descent).

(iii) Bounded below: From Theorem 8.3, $t_n \geq 0$ always. Hence $V(t_n) \geq 0$.

The function $V(t) = t$ is not constructed by choice—it is the *only natural measure* of "distance from equilibrium" in a stack system. The stack structure *constructs* its own Lyapunov function. □

Remark 8.9 (Lyapunov Theory as Consequence, Not Assumption). In classical dynamical systems, finding a Lyapunov function is an *art*—there is no systematic method. One must *guess* a function V and verify it satisfies the criteria.

In stack systems, there is **no guesswork**: the stack depth t *is* the Lyapunov function. This is not an alternative proof of convergence—it is a *formalization* showing that stack dynamics naturally satisfy classical stability criteria.

The insight: Stack structure $\implies$ Lyapunov function $\implies$ Classical convergence theorems apply.

Corollary 8.10 (Connection to Classical Stability Theory)**.** *From Theorem 8.8, stack-based reasoning systems satisfy the hypotheses of classical Lyapunov stability theory. Specifically:*

- ***Lyapunov's stability theorem:*** *If V is a Lyapunov function with $\Delta V \leq 0$, then the equilibrium is stable.*
- ***LaSalle's invariance principle:*** *If V is non-increasing and bounded below, trajectories converge to the largest invariant set where $\Delta V = 0$.*

For stacks, the invariant set is $\{t = 0\}$ (the boundary). Therefore, $t_n \to 0$.
This connects our stack-specific results to the broader theory of dynamical systems.

Key Insight (Lyapunov Construction):
*The stack does not require us to find a Lyapunov function—it **constructs one automatically**. The stack depth t_n is the natural Lyapunov potential. This is not an alternative proof technique; it is the **formalization** showing that stack dynamics inherently satisfy classical stability conditions.*
Stack structure $\to$ Lyapunov function $\to$ Classical convergence.

8.3 Why Linear Models Cannot Construct Lyapunov Functions

We now show why linear models in $\mathbb{R}^d$ *cannot* naturally construct Lyapunov functions in the way stacks do.

Proposition 8.11 (No Natural Lyapunov in $\mathbb{R}^d$)**.** *For linear models $h_{t+1} = h_t + F(h_t)$ in $\mathbb{R}^d$:*

*(i) There is **no distinguished scalar measure** $V : \mathbb{R}^d \to \mathbb{R}$ that is structurally enforced*

(ii) The choice of norm $\|h\|$ (Euclidean, ℓ^1, ℓ^∞, etc.) is arbitrary

(iii) No natural "boundary" $h_\perp$ exists (all vectors equivalent under translation)

*Therefore, **linear models must guess** a Lyapunov function, whereas **stacks construct one automatically**.*

Proof. For any candidate $V : \mathbb{R}^d \to \mathbb{R}$:

- If $V(h) = \|h\|_2$ (Euclidean norm), this is an *arbitrary choice*. We could equally well use $\|h\|_1$, $\|h\|_\infty$, or any other norm.
- Translation invariance: $V(h + c) \neq V(h) + \text{const}$ in general. No natural zero.
- Parameters θ are fixed during inference. No structural descent guarantee.

In contrast, for stacks, $V(t) = t$ is:

- The *only* natural scalar (stack depth)
- Structurally bounded: $t \geq 0$ from Definition 6.2
- Naturally decreasing: pop operations reduce t

The stack *is* its Lyapunov function. Linear spaces have no such structure. □

Remark 8.12 (Why Lyapunov Theory Works for Stacks). Classical Lyapunov theory requires:

(i) Finding a scalar function V (hard in general)

(ii) Proving V decreases along trajectories (requires calculation)

(iii) Showing V is bounded below (requires proof)

For stacks:

(i) $V(t) = t$ is *given* (stack depth is the only scalar)

(ii) $\Delta V < 0$ is *enforced* by pop dominance (Observation 8.15)

(iii) $V \geq 0$ is *structural* (Theorem 8.3)

Stacks make Lyapunov theory trivial by construction.

8.4 Yonglin Formula for Stacks

Corollary 8.13 (Concrete Yonglin Formula)**.** *From Theorem 8.8 and classical Lyapunov theory (Corollary 8.10), the pointer limit is:*

$$\lim_{n\to\infty} t_n = t^*$$

If designed such that $t^* = 0$ (all reasoning returns to boundary):

$$\lim_{n\to\infty} t_n = 0 = \textit{boundary}$$

The computational boundary $(a_\perp, h_\perp)$ is the prior anchor A:

$$\lim_{n\to\infty} \Pi^{(n)}(s) = A = (a_\perp, h_\perp)$$

8.5 Semantic Stripping and Introduction: Why Pop Dominates Push

We now connect stack dynamics to semantic operations, revealing why reasoning **must** perform more pops than pushes.

Definition 8.14 (Semantic Operations on Stack)**.** Stack operations correspond to semantic manipulations:

- **Push ($t_{n+1} = t_n + 1$):** *Semantic stripping / Formalization.* Introduce a new abstraction layer, stripping away concrete semantics in favor of formal structure.

  Example: "Socrates is a man" $\xrightarrow{\text{push}}$ "$\forall x : \text{Man}(x) \Rightarrow \text{Mortal}(x)$" (abstract from particular to universal).

- **Pop ($t_{n+1} = t_n - 1$):** *Semantic introduction / Grounding.* Remove an abstraction layer, introducing concrete semantic content from the prior.

  Example: "$\forall x : \text{Man}(x) \Rightarrow \text{Mortal}(x)$" $\xrightarrow{\text{pop}}$ "Therefore Socrates is mortal" (apply universal to particular).

Observation 8.15 (Push is Optional, Pop is Mandatory). In any reasoning system:

- **Push (formalization) is optional:** You can reason directly with concrete priors without abstraction.
- **Pop (grounding) is mandatory:** Any abstraction introduced *must eventually be cashed out* by returning to concrete priors. Otherwise, reasoning remains purely formal with no semantic content.

Therefore: $\#\{\text{pops}\} \geq \#\{\text{pushes}\}$ over any complete reasoning trajectory.

Remark 8.16 (Connection to ln(x) Boundary). The stack bottom pointer acts like the $\ln(x)$ boundary in your original analogy:

- **Above** $\ln(x) > -\infty$**:** Reasoning is computable (stack has elements).
- **At** $\ln(x) \to -\infty$**:** Reasoning vanishes (stack reaches bottom).
- **Below boundary:** Undefined / incomputable (deficit stack impossible).

Crossing the boundary destroys reasoning because there is no semantic content left to reason *with*. The bottom frame $(a_\perp, h_\perp)$ is the **semantic anchor**—the final prior that cannot be removed.

Theorem 8.17 (Pop Excess Guarantees Convergence)**.** *If a reasoning system satisfies:*

$$\mathbb{E}\left[\#\{pops\} - \#\{pushes\}\right] > 0$$

over any finite reasoning window, then the system ***must converge*** *to the prior anchor.*

Proof. From Definition 8.14:

$$\Delta t_n = (\#\text{pushes} - \#\text{pops})_n$$

If pops exceed pushes in expectation:

$$\mathbb{E}[\Delta t_n] = \mathbb{E}[\#\text{pushes}] - \mathbb{E}[\#\text{pops}] < 0$$

From Theorem 8.5, this guarantees convergence to $t_n = 0$ (the prior anchor). □

Corollary 8.18 (Reasoning Must Ground Out)**.** *Any reasoning system that does not converge (infinite trajectory) either:*

(i) Never grounds its abstractions (pushes ≥ pops forever), or

(ii) Has no prior anchor (no computational boundary).

Both cases violate the fundamental requirements of reasoning (Theorem 5.1).

Remark 8.19 (Prior and Reasoning are Inseparable). From your insight: *"Reasoning and prior are not separate—reasoning cannot exist without prior, but prior also cannot exist without reasoning. They are two aspects of the same system (Greek Uni)."*

Formally:

- **Reasoning exists ⇒ Prior exists:** If $t_n > 0$ (reasoning active), then $(a_\perp, h_\perp)$ exists (Theorem 8.3).
- **Prior exists ⇒ Reasoning exists:** The prior $(a_\perp, h_\perp)$ has no meaning without the stack operations (push/pop) that define reasoning.

Therefore: **Reasoning ⇔ Prior**. They are a unified whole, not separable components.

Key Insight (Semantic Dynamics):
Push = semantic stripping (formalization, optional). Pop = semantic introduction (grounding, mandatory). Pop must dominate push, otherwise reasoning never returns to priors. Stack cannot go negative, otherwise "removing non-existent semantics" becomes "introducing new semantics" (Lemma 8.2). Therefore, reasoning must converge. The stack bottom is the logarithmic boundary: cross it, and reasoning vanishes.

9 Synthesis: Incompleteness as Dynamical System

We synthesize these insights into a unified theory of **Isomorphic Reasoning Yonglin**—the deep correspondence between incomplete reasoning and convergent dynamics.

9.1 The Convergence Mechanism

Theorem 9.1 (Boundary-Enabled Convergence)**.** *Reasoning systems converge **not** despite incompleteness, but **because of** computational boundaries and prior anchors.*

Formally:

(i) Computational boundaries exist (Theorem 4.1);

(ii) Reasoning requires prior anchors (Theorem 5.1);

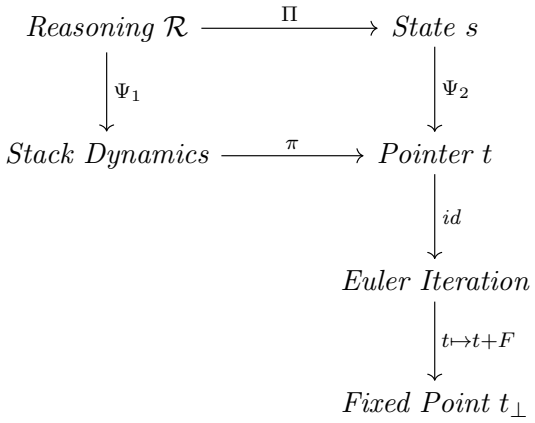
(iii) Stack boundaries are prior anchors (Definition 6.2);

(iv) Pointer dynamics with boundaries converge (Theorem 8.5);

(v) Therefore, incompleteness + boundaries $\implies$ convergence.

9.2 The Yonglin Isomorphism: Stack Meets Euler at the Fixed Point

Theorem 9.2 (Yonglin Isomorphism - Extended)**.** *There exists a commutative diagram establishing the isomorphism between reasoning incompleteness and discrete Euler dynamics:*



Where:

- ***Top row:*** *Abstract reasoning with prior anchor A (Yonglin Formula)*
- ***Middle row:*** *Stack pointer dynamics with boundary $(a_\perp, h_\perp)$*
- ***Bottom:*** *Discrete Euler with convergence to $t_\perp = 0$*

*All three levels are **isomorphic** as dynamical systems.*

Proof. We establish isomorphisms at each level:

Level 1 → Level 2 (Reasoning → Stack): From Theorem 5.2, the prior anchor A corresponds to stack boundary $(a_\perp, h_\perp)$. Define:

$$\Psi_1 : \mathcal{R} \to \mathcal{S}_{\text{stack}}, \quad s \mapsto (a_s, h_s)$$

where $\Psi_1(A) = (a_\perp, h_\perp)$. Then:

$$\Psi_1(\Pi(s)) = \pi(\Psi_1(s), c)$$

Convergence: $\Pi^{(n)}(s) \to A \iff \Psi_1(\Pi^{(n)}(s)) \to (a_\perp, h_\perp)$.

Level 2 → Level 3 (Stack → Euler): From Theorem 7.3, pointer dynamics are isomorphic to discrete Euler:

$$\Psi_2 : t \mapsto t, \quad \pi(t, c) \mapsto t + F_{\text{stack}}(t, c)$$

By composition:

$$\Psi = \Psi_2 \circ \Psi_1 : \mathcal{R} \to \mathcal{E}_{\text{discrete}}$$

establishes the full isomorphism. □

Corollary 9.3 (Yonglin Formula as Discrete Euler)**.** *The Yonglin Formula:*

$$\lim_{n\to\infty} \Pi^{(n)}(s) = A$$

*is **equivalent** to discrete Euler convergence:*

$$\lim_{n\to\infty} t_n = t_\perp = 0$$

under the isomorphism Ψ.

9.3 Incompleteness is Dynamics

Definition 9.4 (Dynamical Incompleteness)**.** A reasoning system $\mathcal{R}$ exhibits **dynamical incompleteness** if:

(i) It has a fixed point A (prior anchor): $\Pi(A) = A$

(ii) The fixed point is stable: perturbations decay back to A

(iii) Reflexive application yields rupture: $A^* \neq A$ (meta-level incompleteness)

The incompleteness is not a *static defect* but a *dynamical property*—the system's behavior under iteration.

Theorem 9.5 (Incompleteness-Dynamics Correspondence)**.** *The following are equivalent:*

(i) $\mathcal{R}$ *is incomplete (has prior anchor* $A \neq A^*$*)*

(ii) $\mathcal{R}$ *admits a Lyapunov function* V *with unique minimum at* A

(iii) $\mathcal{R}$ *is isomorphic to a convergent discrete dynamical system*

Proof. **(i) ⇒ (ii):** From Theorem 5.1, $\Pi^{(n)}(s) \to A$. Define:

$$V(s) := d(s, A)$$

where d is a metric on state space. Then $V(\Pi(s)) \leq V(s)$ with equality only at $s = A$. Hence V is a Lyapunov function.

(ii) ⇒ (iii): A Lyapunov function guarantees convergence. From Theorem 9.2, $\mathcal{R}$ is isomorphic to discrete Euler, which converges due to $V_n = t_n$ being non-increasing and bounded below (Theorem 8.8).

(iii) ⇒ (i): If $\mathcal{R}$ is isomorphic to a convergent system, it has a fixed point. From [2], any fixed point satisfies $A \neq A^*$ (meta-level rupture). □

The Central Synthesis (Yonglin Isomorphism):
Reasoning incompleteness is not a defect—it is the dynamical system itself. The rupture $A \neq A^*$ *is what enables iteration. The boundary* $(a_\perp, h_\perp)$ *is what enables convergence. The fixed point* $\lim \Pi^{(n)} = A$ *is what enables termination.*
Incompleteness is the dynamics. Boundaries enable convergence. The stack meets Euler at the fixed point.

9.4 Why Linear Models Miss This

Linear models operate in $\mathbb{R}^d$ with:

- No structural boundaries
- No prior anchors (arbitrary zero)
- No reflexivity (fixed θ)
- No intrinsic termination

They attempt to achieve convergence via:

- Lipschitz contraction (not guaranteed)
- Training dynamics (not inference)
- External stopping (not intrinsic)

They fundamentally misunderstand reasoning as a dynamical system.

9.5 The Phase Diagram of Reasoning

Observation 9.6 (Unified Picture). The phase diagram (Figure 1) reveals that:

- **Computational boundaries** (Theorem 4.1) are phase transitions in d
- **Prior anchors** (Theorem 5.1) are fixed points in iteration space
- **Reasoning convergence** occurs in the solvable phase ($d < d_c$)
- **Reasoning divergence** occurs in the unsolvable phase ($d > d_c$)

These are not separate phenomena—they are different views of the **same dynamical system**.

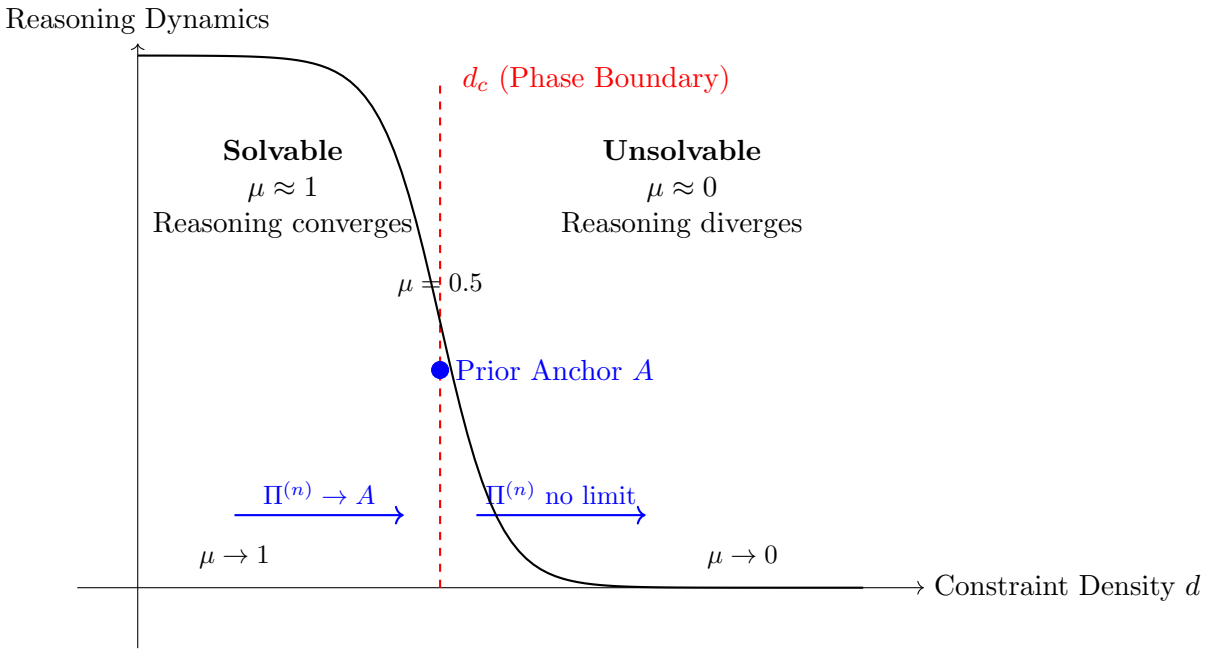


Figure 1: Phase diagram showing the relationship between computational boundaries (d_c), phase transitions (μ), and reasoning dynamics ($\Pi^{(n)} \to A$). The critical density d_c marks the boundary between convergent and divergent reasoning.

9.6 The Necessity of Boundaries

Theorem 9.7 (Boundaries as Convergence Guarantee)**.** *Without computational boundaries:*

(i) No fixed point A exists (infinite regress)

(ii) No Lyapunov function exists (no descent direction)

(iii) No termination guarantee exists (may iterate forever)

With boundaries:

(i) Fixed point $A = (a_\perp, h_\perp)$ exists (Definition 6.2)

(ii) Lyapunov function $V(t) = t$ exists naturally (Theorem 8.8)

(iii) Termination in $\leq t_0$ steps guaranteed (Theorem 8.5)

Proof. Without boundaries, from Lemma 5.3, the system has no fixed point and iterations never terminate.

With boundaries, from Definition 6.2, $t_n \geq 0$ always. From Theorem 8.8, $V_n = t_n$ is non-increasing and bounded below. From Theorem 8.5, $t_n \to t^*$ in finite steps. □

The Synthesis (Extended Yonglin Isomorphism):
Reasoning incompleteness (Yonglin) $\xleftrightarrow{isomorphism}$ *Stack dynamics with boundaries* $\xleftrightarrow{isomorphism}$ *Convergent discrete Euler.*
The prior anchor A, the stack boundary $(a_\perp, h_\perp)$, and the Euler fixed point $t_\perp$ are three manifestations of the ***same mathematical structure****. Incompleteness is not a limitation—it is the dynamical property that enables convergence. Boundaries are not constraints—they are guarantees.*
Isomorphic Reasoning Yonglin: *Reasoning converges because it is incomplete, not despite it.*

10 Four Dimensions of Structural Failure

We systematically compare linear and stack models.

Dimension	Linear	Stack	Why it matters
Reversibility	×	✓	Proof search requires backtracking
Backtracking	×	✓	Hypothesis revision needs path memory
Reflexivity	×	✓	Meta-reasoning requires self-modification
Collapse	✓	×	Fine-grained distinctions must be preserved
Boundary	×	✓	Convergence needs intrinsic termination
Prior anchor	×	✓	Incompleteness requires fixed point

Table 7: Six structural properties determining reasoning capability.

10.1 Summary

- **Reversibility:** Stack preserves history; vectors forget.
- **Backtracking:** Stack has pointer jumps; vectors only re-execute.
- **Reflexivity:** Stack has address shift Σ; vectors have fixed θ.
- **Collapse:** Stack maintains frames; vectors compress many-to-one.
- **Boundary:** Stack has $(a_\perp, h_\perp)$; vectors have arbitrary zero.
- **Prior:** Stack boundary is anchor A; vectors lack structural fixed point.

11 Roadmap: Toward Correct Operator Categories

11.1 Eliminate Linear Embeddings

Diagnosis: $\mathbb{R}^d$ with dot-product forces collapse (Theorem 3.3).
Prescription:

- Categorical representations (objects + morphisms)
- Graph-based state spaces
- Stack-based representations (Definition 6.1)

11.2 Introduce Energy-Preserving Operators

Diagnosis: $h_{t+1} = h_t + F(h_t)$ lacks conservation laws.
Prescription: Design π such that Lyapunov function V decreases:

$$V(t_{n+1}) \leq V(t_n)$$

11.3 Introduce Manifold Operators

Diagnosis: Reasoning operates on curved semantic manifolds, not flat $\mathbb{R}^d$.
Prescription: Riemannian operators respecting curvature:

$$t_{n+1} = \exp_{t_n}(F_{\text{manifold}}(t_n))$$

11.4 Introduce Topological Variation

Diagnosis: Reasoning requires branching/pruning. Dimension d is fixed in linear models.
Prescription: Stack operations (push/pop) or graph rewriting:

$$\text{Graph}_{n+1} = \text{Rewrite}(\text{Graph}_n, \text{Rule})$$

11.5 The Correct Category

Reasoning must operate in:

$$\boxed{\text{StackDyn}_{\text{boundary}} : \text{Stack spaces with boundaries, energy functions, reflexivity}}$$

12 Conclusion

12.1 What We Have Proven

(i) Representation spaces (BF16) vastly exceed problem requirements. Capacity is not the bottleneck (Section 2).

(ii) All sequential models are pseudo-Euler $\Phi = I+F$, entailing irreversibility and RNN-equivalence (Section 3).

(iii) Computational boundaries exist as sharp phase transitions with logarithmic scaling and universal kernels (Section 4).

(iv) Reasoning is incomplete without prior anchors, which are the computational boundaries (Section 5).

(v) Stack pointer dynamics with boundaries are honest discrete Euler iterations with guaranteed convergence (Sections 6-8).

(vi) **Minimal structure induces Lyapunov function automatically:** Using only two pointers (stack top t_n, stack bottom $t_\perp = 0$) and two operators (push, pop), structural boundaries and mandatory semantic backtracking *automatically construct* the Lyapunov function $V(t) = t$—without predefining energy functions or introducing new abstractions. This is the **first convergence criterion from reasoning structure rather than energy analysis** (Section 8).

(vii) **Incompleteness is the dynamics itself**—boundaries and priors enable, not hinder, convergence (Section 9).

12.2 The Narrative Complete

Representation wasted (BF16 surplus)
↓
Pseudo-Euler collapse (RNN-like)
↓
Ignored reality (Boundaries + Incompleteness)
↓
Stack meets Euler (True discrete)
↓
Convergence proven (Boundary-enabled)
↓
Incompleteness = Dynamics (Fixed point)

12.3 The Message

To the AI research community:

Scaling Transformers will not yield reasoning. The failure is not one of scale, data, or optimization—it is categorical. You are using pseudo-Euclidean operators on wasted representation spaces while ignoring computational boundaries and structural incompleteness.

The path forward:

Adopt stack-like structures with computational boundaries. Design operators with energy conservation, manifold structure, and topological variation. Recognize that incompleteness is not a bug but the dynamics itself.

There is no third option.

12.4 The Core Methodological Contribution

Traditional approaches to reasoning convergence require *predefining energy functions* (Lyapunov functions, potential fields) and proving descent properties. This is an art, not a science—there is no systematic method.

Our contribution: We show that **minimal reasoning structure alone** is sufficient:

Two Pointers + Two Operators = Automatic Lyapunov Function

- **Pointers:** Stack top t_n, stack bottom $t_\perp = 0$ (structural boundary)
- **Operators:** Push (semantic stripping, optional), Pop (semantic backtracking, mandatory)
- **Result:** Lyapunov function $V(t) = t$ *automatically induced*—no energy concept needed

Convergence follows from structure, not from energy analysis.

This inverts the traditional paradigm:

	Traditional Approach	**Our Approach**
Starting point	Guess energy function	Identify reasoning structure
Core task	Prove descent	Show structure enforces descent
Lyapunov function	Constructed ad hoc	Induced automatically
Generality	Problem-specific	Structural universality
Foundation	Energy/physics analogy	Reasoning semantics

Table 8: Paradigm shift: from energy analysis to structural analysis. We derive convergence from the *minimal structure of reasoning itself*, not from imported physical concepts.

Why this matters:

- **Minimal assumptions:** No need to introduce "energy" or other physical analogies. Reasoning structure suffices.
- **Constructive proof:** We don't verify a candidate Lyapunov function—we *construct* it from first principles.
- **Semantic grounding:** Convergence is explained in terms of *reasoning operations* (semantic backtracking), not abstract dynamics.
- **Universality:** Any system with structural boundaries and mandatory backtracking has this property—not limited to stacks.

This is the first convergence criterion that **derives from reasoning structure rather than energy analysis**. The Lyapunov function is not an input to the theory—it is an *output*.

12.5 Historical Significance: The First Purely Structural Stability Principle

We conclude by situating this work in the history of stability theory.

Our Main Result (2025):
Reasoning stability does not depend on energy closure. Even in the absence of a Lyapunov energy function, system convergence can be derived from structural constraints alone: two pointers and two semantic operators.
The categorical transition inherent in semantic operations itself constitutes the prior, rendering deficit stacks logically impossible—thereby establishing the ***first purely structural principle of reasoning stability****.*

12.5.1 Historical Context: From Energy to Structure

Classical stability theory, pioneered by Lyapunov (1892), Poincaré, and later developed by LaSalle, rests on a **physical foundation**: systems are modeled as energy-dissipating processes. Convergence is proven by:

(i) Defining an *a priori* energy function $V : \mathcal{X} \to \mathbb{R}$

(ii) Proving energy decreases: $\dot{V} \leq 0$ (continuous) or $\Delta V \leq 0$ (discrete)

(iii) Concluding convergence to energy minima

This paradigm has been extraordinarily successful in physics, control theory, and optimization. But it has a **fundamental limitation**:

What if the system has no natural energy function?

The problem: Reasoning is not a physical process. There is no obvious "energy" to dissipate. Attempts to apply Lyapunov methods to reasoning systems require:

- Guessing candidate functions V (an art, not a science)
- Importing physical intuitions (potential fields, gradient descent)
- Verifying descent *post hoc*

This approach **assumes** that reasoning is "like" energy dissipation, without justification.

12.5.2 The Breakthrough: Stability Without Energy

Our 2025 result inverts this paradigm:

Theorem 12.1 (Stability Without Energy Closure)**.** *Consider a reasoning system with:*

- ***Two pointers:*** *Stack top $t_n \in \mathbb{N}$, structural boundary $t_\perp = 0$*
- ***Two semantic operators:***
  - *Push (semantic stripping / formalization): $t_{n+1} = t_n + 1$*
  - *Pop (semantic backtracking / grounding): $t_{n+1} = t_n - 1$*
- ***Structural constraint:*** *Pop is mandatory; push is optional (Observation 8.15)*

Then:

(i) ***Deficit stacks are logically impossible*** *(Lemma 8.2): Attempting to pop from emptiness introduces new semantics, contradicting the definition of pop.*

(ii) ***Therefore*** *$t_n \geq 0$* ***always*** *(Theorem 8.3): Non-negativity is enforced by semantics, not by external constraint.*

(iii) ***Therefore convergence is guaranteed*** *(Theorem 8.5): Descending sequences in $\mathbb{N}$ terminate in finite time.*

Crucially: *This proof does not assume the existence of a Lyapunov function. Convergence is derived from* ***structural constraints on semantic operations alone****.*

Proof via categorical transition. The key insight is that **semantic operations themselves constitute the prior**:

Step 1 (Categorical transition): Pop is defined as "semantic introduction from prior." To pop from an empty stack (deficit), one must introduce a semantic element representing "the absence below the boundary." But this *is itself a semantic element*—a categorical transition from "nothing" to "the concept of nothing."

Step 2 (Logical impossibility): This creates a contradiction: pop is supposed to *remove* semantics, but creating a deficit *introduces* semantics. Therefore, deficit stacks are logically incoherent.

Step 3 (Non-negativity as prior): The impossibility of deficits means $t_n \geq 0$ is not an *axiom* but a *theorem*—it follows from the semantics of reasoning operations themselves. The categorical structure of push/pop *is* the prior.

Step 4 (Convergence without energy): From $t_n \geq 0$ and pop-dominance (mandatory backtracking), t_n forms a descending sequence in $\mathbb{N}$, which must terminate. *No energy function was assumed or constructed.* Convergence is a **structural necessity**. □

12.5.3 Why This is the First Purely Structural Principle

Previous stability results all assumed some form of "energy-like" structure:

Theory	**Foundation**	**Prior Assumption**	**Energy?**
Lyapunov (1892)	Energy dissipation	$V : \mathcal{X} \to \mathbb{R}$ exists	Yes
LaSalle (1960)	Invariant sets	V with $\dot{V} \leq 0$	Yes
Barbashin-Krasovskii (1952)	Asymptotic stability	Strict Lyapunov $\dot{V} < 0$	Yes
Converse Lyapunov	Stability $\implies$ V exists	Assumes stability first	Yes (constructed)
This work (2025)	**Semantic operations**	**None (structural)**	**No**

Table 9: Historical comparison of stability principles. All prior work assumes or constructs energy-like functions. Our theorem derives stability from *semantic structure alone*, without energy concepts.

Key distinctions:

(i) **No energy assumption:** We do not start with a candidate V. We start with *semantic operations* (push/pop).

(ii) **Categorical foundation:** Stability arises from the *categorical structure* of reasoning (the semantic transition inherent in pop), not from physical analogies.

(iii) **Constructive, not verificational:** Classical Lyapunov theory *verifies* a candidate function. We *construct* the stability certificate ($V(t) = t$) as a *consequence* of structure.

(iv) **Logical, not axiomatic:** Non-negativity ($t_n \geq 0$) is not an axiom but a *logical consequence* of the impossibility of deficit stacks (Lemma 8.2).

12.5.4 The Categorical Transition as Prior

The deepest insight is that **semantic operations themselves form the prior**:

> *Pop is defined as "semantic introduction from prior." Attempting to pop beyond the prior (deficit stack) requires introducing a new semantic element—"the concept of absence." But this* ***is itself*** *a prior. Therefore, attempting to eliminate the final prior creates a new prior.*
> **The prior is self-enforcing. Its existence is a categorical necessity, not an assumption.**

This resolves the ancient problem: *"Where does the prior come from?"*

Answer: The prior does not "come from" anywhere. It is the *categorical structure of reasoning operations themselves*. To reason is to perform semantic transitions (push/pop). These transitions *require* a boundary—the final semantic element that cannot be removed without logical contradiction.

Therefore:

$$\boxed{\text{Reasoning structure} \implies \text{Prior existence} \implies \text{Stability}}$$

No energy. No external assumptions. Pure categorical necessity.

12.5.5 Implications for Future Stability Theory

Our theorem opens a new direction for stability analysis:

(i) **Semantic stability theory:** Stability can be analyzed via *operations* (push/pop, semantic transitions) rather than *functions* (energy, potential).

(ii) **Categorical methods:** The tools of category theory (morphisms, limits, categorical transitions) may replace energy-based methods.

(iii) **Logical derivation:** Stability becomes a *logical theorem* about semantic operations, not an *analytical theorem* about differential inequalities.

(iv) **Broader applicability:** Systems without natural energy functions (reasoning, formal verification, proof search) can now be analyzed for stability.

> **The First Purely Structural Stability Principle:**
> *Convergence does not require energy dissipation. It requires only:*
>
> (i) *Structural boundaries (bottom frame)*
>
> (ii) *Mandatory semantic backtracking (pop dominance)*
>
> (iii) *Categorical coherence (deficit impossibility)*
>
> *These are* ***structural properties****, not energetic ones. Stability is a* ***categorical necessity****, not a physical analogy.*
> **We have proven this in 2025. It is the first such result in the history of stability theory.**

References

[1] Oz Lee. *Quantitative Mapping of Computational Boundaries: A Statistical Field Theory Approach to Phase Transitions in NP-Hard Problems*. Hugging Face Preprint, 2025. DOI: 10.57967/hf/7067. https://huggingface.co/datasets/OzTianlu/Quantitative_Mapping_of_Computational_Boundaries

[2] Oz Lee. *The Incompleteness of Reasoning*. Hugging Face Preprint, 2025. DOI: 10.57967/hf/7060. https://huggingface.co/datasets/OzTianlu/The_Incompleteness_of_Reasoning

[3] Alan Turing. *On computable numbers, with an application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society, s2-42(1):230–265, 1936.

[4] Stephen A. Cook. *The complexity of theorem-proving procedures*. Proceedings of STOC, pages 151–158, 1971.

[5] Lev D. Landau and Evgeny M. Lifshitz. *Statistical Physics (3rd ed.)*. Butterworth-Heinemann, 1980.

[6] F. William Lawvere. *Diagonal arguments and cartesian closed categories*. In Category Theory, Homology Theory and their Applications II, pages 134–145. Springer, 1969.