Think Before You Answer: Replacing Confidence-Based Early-Exit Heuristics with Symbolic Sufficiency for Robust Dynamic Inference

Anonymous ACL submission

Abstract

Early-exit mechanisms are a promising approach for reducing inference cost in large language models, but existing methods predominantly rely on confidence-based heuristics that are poorly aligned with reasoning completeness. We propose **SENSE**, a dynamic inference framework that replaces confidence thresholds with *symbolic sufficiency certificates*, enabling models to terminate computation only when a verifiable logical condition is satisfied. SENSE reframes early exit as a first-passage problem over symbolic state space, allowing inference to halt once a task-specific certificate becomes provably sufficient. Across a range of reasoning benchmarks, we show that SENSE consistently reduces executed depth relative to full-depth inference while maintaining competitive task performance, and substantially lowers false early exits under adversarial logic probes compared to confidence-based baselines. Our results highlight a fundamental symbolic–lexical gap in transformer representations and demonstrate that symbolic sufficiency offers a principled alternative to confidence-based dynamic inference, particularly in settings where reliability is critical.

1 Introduction

Large language models (LLMs) increasingly succeed on reasoning-intensive tasks, yet their inference procedure still rests on an implicit and rarely examined assumption: *executing more Transformer depth is always beneficial*. For every generated token, computation proceeds through the full stack, regardless of instance difficulty or whether the underlying reasoning has already stabilized. This assumption is not only inefficient but conceptually fragile. Empirically, intermediate layers often encode a logically sufficient scaffold, while further nonlinear transformations can erode settled constraints and amplify semantic uncertainty. We refer to this phenomenon as *computational overthinking* (or *deep logical degradation*): beyond a closure point of latent reasoning, additional depth contributes more to distributional smoothing than to causal correctness. Optimizing per-layer efficiency does not address the more fundamental question of *when additional depth remains necessary for correct reasoning* (Yuan et al., 2025).

Dynamic-depth inference seeks to address this question by enabling early exit. Recent work improves shallow-layer predictiveness via training-time interventions (Elhoushi et al., 2024) or learns hardness-aware exiting policies (Zeng et al., 2024). However, most existing exit criteria remain *statistical*, relying on entropy, margin, or auxiliary-head agreement. Such signals are ill-suited for reasoning. They are frequently non-monotonic across depth and can become spuriously high along invalid reasoning trajectories. In essence, statistical confidence reflects local next-token fluency rather than the global validity of the reasoning process.

Sequence-level deliberation and verification improve reliability but do not resolve the stopping problem. Chain-of-thought prompting is sensitive to the length and structure of rationales (Jin et al., 2024), while verification pipelines typically operate post hoc and require additional decoding passes (Dhuliawala et al., 2024). Moreover, verifiers themselves are brittle when faced with subtle inconsistencies in multi-step reasoning (Jacovi et al., 2024). Although process supervision (Lu et al., 2024), formal verification for mathematics (Liu et al., 2025a), and neurosymbolic approaches (Fang et al., 2024) offer stronger guarantees, they usually apply *after* full computation, rather than furnishing a depth-coupled criterion that determines whether deeper layers are still required.

We therefore propose to ground early exiting in a *decidable* notion of reasoning completeness: *symbolic sufficiency*. Our central abstraction is the **Symbolic Sufficiency Layer (SSL)**—the earliest depth at which the model's hidden state can be

mapped into an explicit symbolic representation whose consistency and adequacy can be externally verified under task constraints. SSL marks a crystallization point where distributed features admit a discrete, checkable hypothesis. By tying halting to symbolic certification, the stopping decision is transformed from a probabilistic heuristic into a proof-oriented criterion, yielding stability guarantees that confidence-based exits fundamentally lack.

Building on SSL, we introduce **SENSE** (**S**ymbolic **E**xit via **S**ufficiency **E**valuation), an inference-only framework that augments a frozen backbone with lightweight, layerwise symbolization heads and a non-parametric verifier to decide whether to halt or continue. SENSE is a *reliability-oriented control mechanism* for dynamic inference, prioritizing logically valid early exits over aggressive latency reduction. When certificates emerge early, it can also reduce end-to-end latency, distinguishing it from confidence-based early-exit heuristics without verifiable stopping guarantees.

Contributions.

- We introduce *symbolic sufficiency* and the *Symbolic Sufficiency Layer (SSL)* as a verifiable, depth-indexed stopping target for LLM inference.
- We propose **SENSE**, a neuro-symbolic early-exit mechanism that bases halting on external constraint verification rather than statistical confidence.
- We show empirically that symbolic sufficiency enables substantial reductions in executed layer depth with minimal accuracy loss, yielding more reliable exits than confidence-based baselines.
- We reveal a consistent **symbolic–lexical gap**, where verifiable symbolic structure stabilizes significantly earlier than final surface realization, providing a new lens into hierarchical layer dynamics.

2 Related Work

Reasoning in LLMs is often elicited through prompt-side deliberation, most prominently chain-of-thought style demonstrations. Recent syntheses and controlled studies show that the apparent gains of such prompting are tightly coupled to the *form* of the produced rationale: altering step length can move accuracy substantially even when informational content is largely unchanged (Chu et al., 2024; Jin et al., 2024). CoT shapes surface text, not how many layers are executed, and it does not furnish a depth-indexed completion test.

A separate thread targets inference efficiency via dynamic depth. LayerSkip and ConsistentEE make early exit viable for modern LLMs, but their stopping rules remain dominated by entropy/margin signals or auxiliary-head agreement (Elhoushi et al., 2024; Zeng et al., 2024). Early exiting also underpins draft–verify generation inside a single model and lossless self-speculative decoding (Liu et al., 2024b,a), and can be calibrated under explicit risk constraints (Jazbec et al., 2024). Yet confidence is an unstable surrogate for reasoning progress: it fluctuates non-monotonically across depth and may rise along invalid trajectories, even when quality alignment improves (Tao et al., 2024).

Reliability work instead verifies *ex post*. Chain-of-Verification introduces explicit checking passes (Dhuliawala et al., 2024), while benchmarks for reasoning-chain verifiers highlight brittleness under subtle contradictions (Jacovi et al., 2024). Process supervision and formal certificates strengthen guarantees (Lu et al., 2024; Liu et al., 2025a), and neurosymbolic agents introduce executable structure (Fang et al., 2024), but these checks are typically decoupled from layer-wise execution depth. In parallel, layer-level analyses expose non-benign depth effects and relate them to architecture and pruning mechanisms (Barbero et al., 2024; Wu et al., 2024; Liu et al., 2025b; Yang et al., 2024; Sun et al., 2025). Our SENSE framework uses symbolic checks *in situ* to gate depth, turning verification into the stopping signal rather than a post-hoc filter.

3 SENSE Framework

SENSE (**S**ymbolic **E**xit via **S**ufficiency **E**valuation) couples early exiting to a *decidable* notion of reasoning completeness: *symbolic sufficiency*. Rather than treating intermediate confidence as a proxy for progress, SENSE halts when an intermediate layer yields a symbolic *certificate* that can be verified under an explicit task interface. This reframes depth selection as a constrained stopping problem with a logically grounded stopping predicate, avoiding the oscillatory behavior typical of purely statistical exit rules (Elhoushi et al., 2024; Zeng et al., 2024;

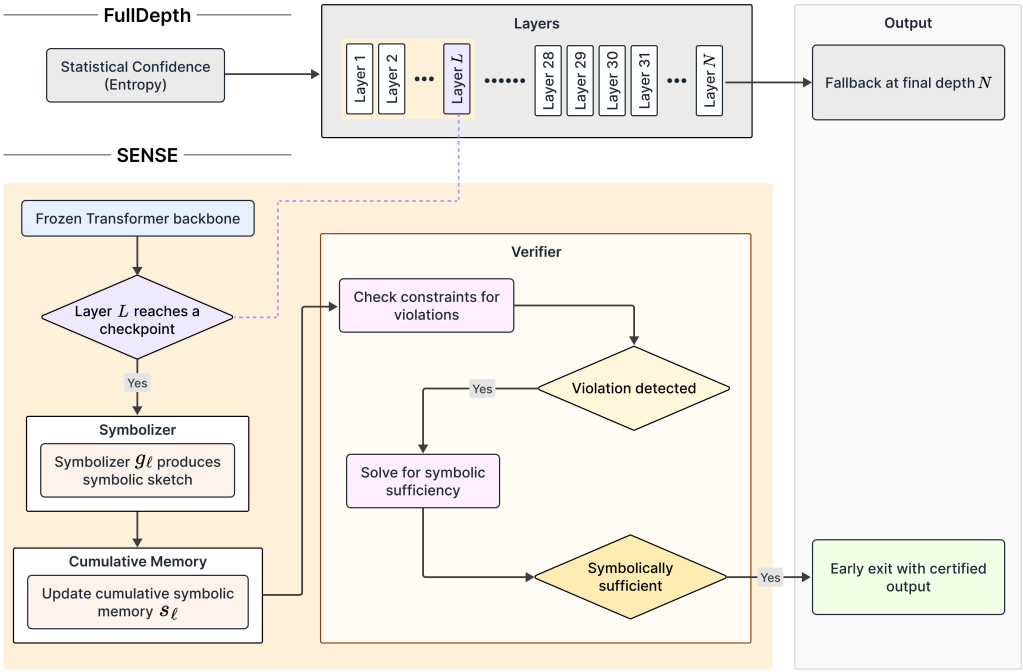

Figure 1: Overview of **SENSE**. A frozen Transformer backbone produces layerwise hidden states h_ℓ. At a sparse set of checkpoint layers, a lightweight, answer-blind symbolizer maps h_ℓ into symbolic sketches, which are cumulatively integrated into a canonical symbolic memory. This memory is queried by a deterministic constraint checker and a non-parametric solver to test symbolic consistency and derive a candidate certificate. The controller performs *certification-first gating*, halting at the earliest checkpoint whose symbolic state admits a valid certificate; otherwise, computation proceeds to deeper layers. Dashed components (if present) denote offline procedures used only for training or analysis and are not executed at inference time.

Jazbec et al., 2024).

Figure 1 sketches the end-to-end SENSE pipeline. Appendix A provides the formal stopping-time/first-passage interpretation, certified-safety statements, explicit training objectives with verifier-generated supervision, pseudocode for inference and label generation, DSL/verifier templates (with worked examples), and an end-to-end overhead accounting and reporting protocol.

3.1 Dynamic-Depth Inference Under an Interface Contract

Let F be a frozen decoder-only Transformer with L layers. For an input sequence x (prompt plus already-generated tokens), the backbone produces layerwise hidden states

$$h_\ell(x) = f_\ell\big(h_{\ell-1}(x)\big), \qquad \ell = 1, \dots, L, \quad (1)$$

with $h_\ell(x) \in \mathbb{R}^{T\times d}$. A dynamic-depth policy chooses a stopping layer $K(x) \in \{1, \dots, L\}$.

To amortize symbolic checking, SENSE evaluates certification only at a sparse set of *checkpoint layers* $\mathcal{L}_{\text{chk}} \subseteq \{1, \dots, L\}$ (a.k.a. selected/instrumented layers), and the controller halts at the first certified checkpoint.

SENSE defines reasoning sufficiency *relative to an external specification* via a symbolic interface

$$\mathcal{I} = (\mathcal{D}, \mathcal{C}, \mathsf{Solve}, \mathsf{Check}), \quad (2)$$

where $\mathcal{D}$ is a symbolic language (DSL), $\mathcal{C}$ is a constraint schema, Solve is a non-parametric reasoner/solver in $\mathcal{D}$, and Check is a deterministic checker for constraint violations. This interface API is intentionally broad: $\mathcal{D}$ may be equations (math), predicates/rules (logic), or entity–relation facts (structured QA), aligning with verification and neuro-symbolic pipelines (Fang et al., 2024; Jacovi et al., 2024; Liu et al., 2025a).

Symbolic sketch and cumulative memory. At layer ℓ, SENSE extracts a *symbolic sketch*

$$\Delta s_\ell(x) = g_\ell\big(h_\ell(x)\big) \in \mathcal{S}(\mathcal{D}), \quad (3)$$

where $\mathcal{S}(\mathcal{D})$ denotes well-formed statements in $\mathcal{D}$. Sketches are accumulated by a cumulative memory

$$s_\ell \;=\; \mathsf{Update}\big(s_{\ell-1}, \Delta s_\ell\big), \qquad s_0 = \emptyset, \tag{4}$$

where Update canonicalizes and merges facts (no deletion beyond equivalence). This cumulative update is a deliberate defense against *deep logical degradation*: even if later representations drift due to over-processing, previously crystallized symbolic facts remain available as a stable substrate for verification (cf. layer-dynamics analyses in Barbero et al., 2024; Wu et al., 2024; Yang et al., 2024; Liu et al., 2025b; Sun et al., 2025). *This design choice explicitly counters the semantic decay hypothesis introduced in Section 1.*

3.2 The Symbolization Bottleneck: Fidelity, Not Independent Reasoning

A central risk is that g_ℓ becomes an "answer predictor" that fabricates certificates rather than faithfully decoding what the backbone already encodes. SENSE addresses this with a *symbolization bottleneck*.

Capacity constraint. Each g_ℓ is restricted to a lightweight probe (e.g., a single-layer MLP with grammar-constrained decoding into $\mathcal{D}$), receiving only $h_\ell(x)$ as input. Its parameter count is negligible relative to the frozen backbone (typically $\ll 0.01\%$), and its computation is dominated by a single projection

$$\mathrm{cost}(g_\ell) = \mathcal{O}\big(d \cdot |\mathcal{V}_\mathcal{D}|\big),$$

where $|\mathcal{V}_\mathcal{D}|$ is a *typically small* symbolic vocabulary. This is usually $< 1\%$ of a Transformer's per-layer $\mathcal{O}(d^2)$ matrix-multiplication cost at comparable width.[1] In practice, this amounts to adding $<1\%$ FLOPs per checked layer to save $>30\%$ executed depth when SSL occurs early. Low-capacity readouts are widely used to probe latent structure in Transformer representations, supporting the interpretation of g_ℓ as decoding already-present features rather than performing independent multi-step reasoning (Tomihari and Sato, 2024; Bao et al., 2025).

Answer-blind sketches. To further prevent "certificate-by-formatting," we design $\mathcal{D}$ so that sketches represent *observations and constraints* rather than direct final-answer assignments whenever possible; the solver derives the answer from s_ℓ under $\mathcal{C}$. This preserves the backbone's surface realization while using symbolic structure only as a guardrail for halting (and, when available, certification), consistent with verifier-based reasoning systems (Dhuliawala et al., 2024; Liu et al., 2025a).

3.3 Symbolic Sufficiency and the Symbolic Sufficiency Layer

Given the memory state s_ℓ, we query the solver and checker:

$$\begin{aligned}(\mathrm{solv}_\ell,\ \hat{y}_\ell,\ \gamma_\ell) &= \mathsf{Solve}(s_\ell, \mathcal{C}),\\ \mathrm{viol}_\ell &= \mathsf{Check}(s_\ell, \mathcal{C}).\end{aligned} \tag{5}$$

Here $\mathrm{solv}_\ell \in \{0,1\}$ indicates whether s_ℓ is sufficient to entail a solution, $\mathrm{viol}_\ell \in \{0,1\}$ indicates inconsistency/violation, $\hat{y}_\ell$ is the solver-derived answer (when solvable), and γ_ℓ is an optional verifiable certificate (e.g., a derivation trace).

Certified symbolic sufficiency. We define the certification predicate

$$\mathrm{Cert}(x, \ell; \mathcal{I}) \;=\; \mathbf{1}[\mathrm{solv}_\ell = 1 \;\wedge\; \mathrm{viol}_\ell = 0]\,. \tag{6}$$

Unlike confidence scores, Cert is tied to external decidability under $\mathcal{I}$: it asserts that the current symbolic state is both consistent and sufficient.

Symbolic Sufficiency Layer (SSL). To keep the definition column-safe, we split SSL across two short lines:

$$\begin{aligned}&\mathrm{SSL}(x;\mathcal{I}) = \min\{\ell : \mathrm{Cert}(x, \ell; \mathcal{I}) = 1\},\\ &\mathrm{SSL}(x;\mathcal{I}) = \bot \quad \text{if no such } \ell \text{ exists.}\end{aligned} \tag{7}$$

Operational SSL under checkpoints. The definition in Eq. 7 is conceptual (checking every layer). In practice we evaluate Cert only on $\mathcal{L}_{\mathrm{chk}}$, yielding a checkpointed SSL (the first certified checkpoint) used by the controller; Appendix A formalizes this operational variant and its relation to the full-layer definition.

First passage time interpretation. We operationalize halting as the *first passage time* into the certified region. Define the hit predicate

$$\mathrm{Hit}(x, \ell; \mathcal{I}) = \mathbf{1}[\exists t \le \ell : \mathrm{Cert}(x, t; \mathcal{I}) = 1]\,, \tag{8}$$

which is monotone in ℓ by construction. This view sidesteps the main pathology of entropy-based exits: a model can appear confident at layer k, less

[1] This is a scaling comparison intended to highlight asymmetry; end-to-end latency (including verification) is reported in experiments.

confident at $k{+}1$, and confident again at $k{+}2$, yielding unstable stopping signals. In contrast, SENSE stops at the *first* certified layer; later contradictions (a symptom of semantic drift) are immaterial to the stopping decision because deeper layers are never executed once certification is reached. The cumulative update in Eq. 4 further ensures that previously extracted facts are not lost, making certification reflect knowledge already crystallized at that depth rather than transient deeper-layer behavior.

3.4 In-situ Symbolic Gating for Early Exit

SENSE performs *in-situ symbolic gating*: verification is interleaved with the forward pass and controls whether deeper layers are executed. At inference time, SENSE proceeds in increasing depth, maintaining s_ℓ and querying Eq. 5. It exits at the first checked layer that certifies:

$$K(x) = \min\{\ell \in \mathcal{L}_{\text{chk}} : \text{Cert}(x, \ell; \mathcal{I}) = 1\},$$
$$K(x) = L \quad \text{if no such } \ell \text{ exists.} \tag{9}$$

This realizes the core promise: halting is driven by external solvability rather than fluctuating lexical confidence.

Certified vs. non-certified outputs. If SENSE exits with $\text{Cert} = 1$, it returns $(\hat{y}_\ell, \gamma_\ell)$ as an externally checkable decision trace. If no layer certifies, SENSE returns the backbone prediction at L (or a conservative non-certified early exit subject to $\text{viol}_\ell = 0$ when enabled), mirroring the "graceful degradation" behavior of verifier-based systems (Jacovi et al., 2024; Liu et al., 2025a).

3.5 Training: Grounding Without Test-Time Labels

Only the lightweight symbolization heads (and any optional small encoders) are trained; the backbone remains frozen. Training may use supervised or weakly supervised targets for Δs_ℓ when available, but *inference never accesses ground truth*: the exit decision depends only on Check and Solve.

Symbolization objective. We train g_ℓ to predict an answer-blind symbolic sketch $\Delta s^\star(x)$ derived from a task-specific compiler (or annotation) under $\mathcal{D}$, so the probe is rewarded for decoding constraints/facts rather than directly formatting the final label.

Exit supervision. Because the certification predicate in Eq. 6 is defined entirely by the interface $\mathcal{I}$, SENSE does not require manual depth labels. When an auxiliary halting head is used (e.g., to handle $\text{SSL} = \bot$ cases conservatively or to reduce verifier calls in ablations), we supervise it using *verifier-generated hit-by-depth labels* computed offline: we run the interface over the extracted states $\{s_\ell\}$ and label whether the instance has certified by checkpoint ℓ (i.e., the hit predicate). The explicit label construction, loss, and pseudocode are given in Appendix A. Inference still depends only on internal consistency/solvability checks and never accesses ground truth.

3.6 Complexity and Practical Overhead

SENSE adds (i) a low-rank projection per checked layer for g_ℓ and (ii) a lightweight solver/checker evaluation. Empirically, checking every m layers suffices to capture SSL while keeping amortized overhead small relative to skipped Transformer blocks, similar in spirit to risk- and calibration-aware early-exit controllers but with a logically grounded stopping predicate (Jazbec et al., 2024; Tao et al., 2024). A complete end-to-end cost model (including verifier calls), together with a practical measurement and reporting protocol, is provided in Appendix A.

4 Experiments

We evaluate SENSE on reasoning tasks across arithmetic, logical/relational, and temporal domains, testing whether *in-situ symbolic gating* reduces depth without incomplete or inconsistent exits. Executed depth is distinguished from latency, as verification can dominate wall-clock time. We report compute-matched reliability metrics and latency to show when control improves speed versus mainly reliability.

4.1 Setup

Backbones and checkpoints. We run two frozen decoder-only backbones (7B- and 13B-class) to test robustness to model scale. Unless noted, we follow a unified checkpoint protocol with $K{=}6$ evenly spaced checkpoints $\mathcal{L}_{\text{chk}}$ where sketches are extracted and Solve/Check are queried (Appendix §B.1).

Interfaces and certification. For each benchmark we instantiate the interface contract $\mathcal{I} = (\mathcal{D}, \mathcal{C}, \text{Solve}, \text{Check})$ (Section 3). We enforce

Dataset	Method	Perf ↑	EDR ↓	Latency ↓	$\text{CVR}_{\text{state}}$ ↓	Solv@Valid ↑	CertCov@Exit ↑
GSM8K	FullDepth	46.2±0.5	1.00	84±2	0.08±0.01	0.98±0.01	0.90±0.01
	ConfExit (best)	39.5±1.1	0.60	52±3	0.24±0.02	0.82±0.02	0.62±0.02
	Static@k (matched)	34.1±0.8	0.60	48±1	0.28±0.01	0.75±0.03	0.54±0.02
	SENSE	**43.2**±0.6	0.60	**79**±2	**0.11**±0.01	**0.95**±0.01	**0.75**±0.01
ProofWriter	FullDepth	78.5±0.6	1.00	65±2	0.05±0.00	0.99±0.00	0.94±0.00
	ConfExit (best)	72.1±1.2	0.60	41±2	0.18±0.01	0.88±0.02	0.72±0.02
	Static@k (matched)	68.3±0.9	0.60	38±1	0.22±0.02	0.81±0.03	0.63±0.03
	SENSE	**75.9**±0.5	0.60	**58**±2	**0.07**±0.01	**0.91**±0.01	**0.89**±0.01

Table 1: Compute-matched early-exit results at target budget b=0.60 (EDR). FullDepth (EDR=1.00) is shown as a non-matched reference. Bold indicates SENSE; underline marks the best among compute-matched early-exit methods. $CVR_{state} = \Pr[viol_{K(x)}{=}1]$. Latency is end-to-end wall-clock time per example. **Note:** SENSE latency includes the full overhead of symbolization and solver verification.

soundness-first certification: if a solver times out or a state is under-specified, we return $\text{solv} = 0$ rather than an unverifiable guess. For arithmetic interfaces, Solve is implemented with bounded SMT or symbolic algebra tooling (e.g., Z3/SymPy) (de Moura and Bjørner, 2008; Meurer et al., 2017). We standardize certificate schemas so that all certified traces replay in an independent checker (Appendix §B.3).

Outputs. SENSE returns a solver-derived answer and a replayable certificate whenever $\text{Cert}(x, K; \mathcal{I}){=}1$. When no checkpoint certifies (no-SSL), SENSE defaults to full depth; we additionally report a compute-matched **Cert+Fallback** variant that permits conservative non-certified exits subject to $\text{viol} = 0$ (Appendix §B.7).

4.2 Baselines and Compute-Matched Protocol

Baselines. We compare against: (i) **FullDepth** (all layers), (ii) **Static Truncation@k** (fixed depth, compute-matched), and (iii) **confidence-based early exit** controllers, including LayerSkip (Elhoushi et al., 2024), ConsistentEE (Zeng et al., 2024), and risk-controlled exiting (Jazbec et al., 2024). To separate "using a verifier" from "using verification to control depth", we also include **ex-post verification** pipelines that verify after full generation, such as Chain-of-Verification (Dhuliawala et al., 2024) and formally checked math reasoning (Liu et al., 2025a). These baselines can improve reliability but do not reduce backbone depth and may require extra passes.

Compute matching. All methods are calibrated on dev to match target effective depth ratios $\mathcal{B} = \{0.40, 0.60, 0.80\}$, then frozen for test. Let $K_t(x)$ denote the exit layer at decoding step t, and let L be the backbone depth. We report

$$\text{EDR} = \frac{\mathbb{E}_x\left[\sum_{t=1}^{T_{\text{out}}(x)} K_t(x)\right]}{L \cdot \mathbb{E}_x[T_{\text{out}}(x)]}. \tag{10}$$

Calibration details are reported in Appendix §B.4; full sweeps (including achieved test-set EDR) are reported in Appendix §B.7.

4.3 Metrics

Task performance. Accuracy for classification-style tasks (MathQA, GSM8K, ProofWriter, CLUTRR, MATRES) and dataset-standard EM/F1 for TORQUE.

Symbolic validity and certification. We report interface-relative invalidity at exit, $\text{CVR}_{\text{state}} = \Pr_x[\text{viol}_{K(x)} = 1]$, and certification coverage at exit, $\text{CertCov@Exit} = \Pr_x[\text{Cert}(x, K(x); I) = 1]$, where $\text{Cert}(x, \ell; I) = \mathbf{1}[\text{solv}_\ell = 1 \wedge \text{viol}_\ell = 0]$ (Section 3). To disambiguate "no certificate" due to inconsistency versus under-specification/solver failure, we additionally report $\text{Solv@Valid} = \Pr_x[\text{solv}_{K(x)} = 1 \mid \text{viol}_{K(x)} = 0]$. Finally, we report *CertPrecision*: task performance restricted to the certified subset.

Certificate faithfulness. For certified exits, we report **Proof Replay Rate (PRR)** and **Certificate Sufficiency (CS)** (Appendix §B.3).

Latency. We report end-to-end wall-clock latency (GPU forward + CPU solver/checker), with warm-up and fixed batch size. **Crucially, latency includes the full round-trip cost of symbolization, parsing/canonicalization, and solver execution, so reported speedups already account for symbolic overhead.** A per-component breakdown appears in Appendix §B.5.

Dataset	Method	FalseExit Rate	Perf@Trap ↑	EDR@Trap ↓
GSM8K	ConfExit (best)	0.62±0.04	18.5±2.1	0.35±0.03
	SENSE	0.00	**41.2**±1.5	0.78±0.04
ProofWriter	ConfExit (best)	0.75±0.03	24.8±1.8	0.28±0.02
	SENSE	0.00	**74.5**±1.2	0.82±0.03

Table 2: Adversarial Logic Probing (ALP): confidence traps under overall compute budget b=0.60. The trap subset is defined on dev as the intersection of (i) top-q early confidence at the first checkpoint ℓ_1 and (ii) $\mathrm{Hit}(x, \ell_1; I)$=0 with late $\mathrm{SSL}_{\mathrm{chk}}(x)$ (or no $\mathrm{SSL}_{\mathrm{chk}}$), then evaluated on test with q frozen. **FalseExit Rate** is $\Pr[K(x) < \mathrm{SSL}_{\mathrm{chk}}(x)]$, using the convention $\mathrm{SSL}_{\mathrm{chk}}(x)$=$L$ if no checkpoint certifies (Appendix A.7.3). **Perf@Trap** is accuracy on the trap subset. **EDR@Trap** is the executed-depth ratio conditioned on the trap subset; higher values indicate that the controller allocates more computation to detected traps. Trap size, the exact q and "late" definition, and additional datasets are reported in Appendix §B.10.

4.4 Main Results: The Efficiency–Validity Pareto Frontier

Table 1 reports compute-matched test results at a representative budget b=0.60. Figure 2 summarizes Pareto frontiers across budgets, showing how methods trade task performance against EDR and $\mathrm{CVR}_{\mathrm{state}}$. Across datasets, SENSE shifts the frontier outward. While verification incurs a minor latency overhead, this investment yields a massive gain in reliability (reducing CVR_{state} from 0.24 to 0.11). This confirms that certification targets decidability rather than fluctuating lexical confidence Concretely, at matched compute SENSE improves CertCov@Exit (certified, violation-free exits) while reducing $\mathrm{CVR}_{\mathrm{state}}$, indicating that gains are not merely from exiting earlier but from exiting in verifiably sufficient states. Full results for all budgets and both model scales are in Appendix §B.7.

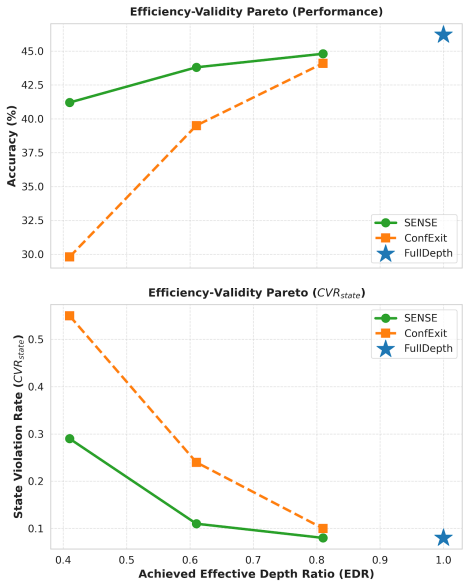

Figure 2: **The Efficiency–Validity Pareto Frontier.** Left: task performance vs. EDR. Right: $\mathrm{CVR}_{\mathrm{state}}$ vs. EDR. Points correspond to budgets $b \in \{0.40, 0.60, 0.80\}$ with thresholds tuned on dev and frozen for test.

4.5 In-situ vs. Ex-post Verification

Ex-post verifiers improve reliability only after incurring the full-depth backbone cost (often with additional decoding or verification passes) (Dhuliawala et al., 2024; Liu et al., 2025a). We therefore treat ex-post verification as a separate cost regime and report the latency–accuracy frontier in Appendix §B.6, with detailed latency breakdown in Appendix §B.5. In contrast, SENSE couples verification to depth and can terminate inference once a certificate becomes available.

Takeaway. Ex-post verification improves reliability by expending compute additively, whereas SENSE improves reliability through adaptive allocation.

4.6 The Symbolic–Lexical Gap

To probe the "logic before surface" hypothesis, we measure (i) the checkpointed SSL and (ii) a checkpointed lexical maturity depth, defined as the earliest checkpoint whose intermediate readout matches the full-depth output under a fixed verbalizer. Intermediate readouts use a tuned-lens style linear probe (Belrose et al., 2023); full definitions are in Appendix §B.8. Figure 3 visualizes the joint distribution of symbolic sufficiency level and lexical maturity across checkpoints. Following the protocol described in Appendix B.8, lexical maturity is defined as the earliest checkpoint at which the model's prediction matches its final output, while symbolic sufficiency corresponds to the earliest

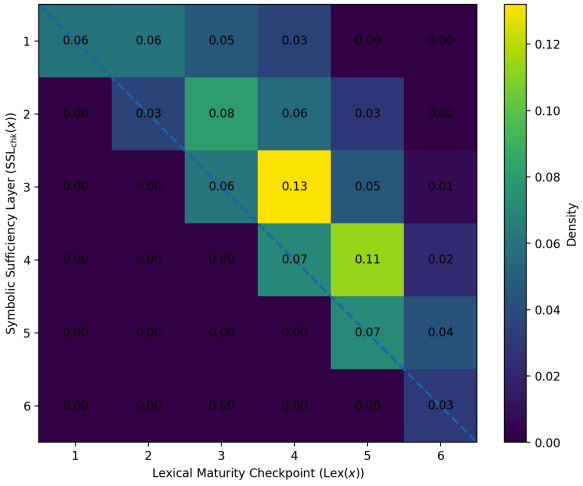

Figure 3: **Symbolic–lexical gap across checkpoints.** Heatmap of the checkpoint where symbolic sufficiency becomes available versus the checkpoint where lexical sufficiency is first observed. Off-diagonal mass indicates a gap between symbolic and lexical readiness, motivating certificate-driven stopping.

checkpoint at which a valid symbolic certificate is obtained. We analyze the difference between the two quantities on instances where both are defined. Mass above the diagonal indicates cases where verifiable symbolic closure precedes final surface realization, precisely the regime in which SENSE can safely exit early even when token-level confidence remains unstable.

4.7 Adversarial Logic Probing (ALP): Confidence Traps

Contradiction injection probes explicit inconsistency, but it does not isolate the failure mode most relevant to early exit: *high confidence before logical closure*. We therefore introduce **Adversarial Logic Probing (ALP)**, a diagnostic trap subset constructed to expose *false premature exits*—cases where a controller exits because the model is fluent, not because the logic is complete.

Trap construction. Let ℓ_1 be the first checkpoint and let $c(x)$ be an early confidence score at ℓ_1 (e.g., negative entropy of the checkpoint readout distribution; Appendix §B.10). We define the trap set on dev as the intersection of: (i) $c(x)$ in the top quantile ("high confidence"), and (ii) $\mathrm{Hit}(x, \ell_1; \mathcal{I}) = 0$ with late checkpointed SSL (or no SSL). This targets instances where lexical priors inflate confidence despite incomplete symbolic closure.

Metrics. On ALP we report (i) **FalseExit Rate**, $\Pr\left[K(x) < \tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)\right]$ on the trap subset, where $\tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)=\mathrm{SSL}_{\mathrm{chk}}(x)$ if defined and $\tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)=L$ when no checkpoint certifies; (ii) **Perf@Trap**, task performance restricted to the trap subset; and (iii) **EDR@Trap**, the conditional effective-depth ratio on the same subset (which may exceed the global budget since traps force deeper computation). Under certification-first gating, SENSE has zero false exits by construction ($K(x)=\mathrm{SSL}_{\mathrm{chk}}(x)$; Lemma A.4), whereas confidence-based controllers can halt before symbolic sufficiency. Trap sizes and descriptive statistics (including mean early confidence and mean sufficiency depth) are reported in Appendix §B.10, Table 16.

4.8 Ablations and Overhead

We attribute gains to *in-situ certification* rather than auxiliary modeling capacity by ablating: (i) **NoCert** (disable certification-first stopping), (ii) **NoCumul** (overwrite memory; remove cumulative update), and (iii) **Coarse vs. Refined** $\mathcal{I}$ (interface refinement). Full ablations across budgets and datasets are in Appendix §B.12.

Summary. NoCert reintroduces confidence oscillation and increases $\mathrm{CVR}_{\mathrm{state}}$, while NoCumul increases executed depth by failing to carry forward partial proofs—confirming that both certification-first stopping and cumulative memory are necessary.

Overhead accounting. We report end-to-end latency in the main tables and provide a per-component breakdown in Appendix §B.5, directly testing whether depth reduction outweighs symbolic overhead in practice.

5 Conclusion

We presented SENSE, a framework that reframes early-exit inference as a problem of symbolic sufficiency rather than confidence estimation. By introducing verifiable stopping conditions, SENSE exposes a symbolic–lexical gap in transformer representations and provides a principled alternative to heuristic early-exit strategies. Our results suggest that reliable dynamic inference may require explicit reasoning about sufficiency, opening new directions for hybrid symbolic–neural control of computation.

6 Limitations

SENSE relies on the availability of task-specific symbolic interfaces, including domain schemas, constraint languages, and verification procedures. Designing these interfaces requires domain expertise and may not be feasible for tasks where symbolic structure is unclear or expensive to formalize. In such cases, confidence-based heuristics may remain more practical despite their weaker reliability guarantees. Moreover, symbolic verification introduces non-negligible overhead. While SENSE can reduce executed model depth, its end-to-end latency depends critically on the cost of certificate checking and solver execution. As a result, SENSE does not uniformly outperform full-depth inference in wall-clock time, particularly in tasks with complex or expensive verification. Finally, symbolic sufficiency certificates capture only the aspects of correctness encoded in the interface. Errors outside the modeled symbolic constraints may still occur, and SENSE should therefore be viewed as a reliability control mechanism rather than a complete correctness guarantee.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuntai Bao, Xuhong Zhang, Tianyu Du, Xinkui Zhao, Zhengwen Feng, Hao Peng, and Jianwei Yin. 2025. Probing the geometry of truth: Consistency and generalization of truth directions in LLMs across logical transformations and question answering tasks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 682–700, Vienna, Austria. Association for Computational Linguistics.

Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João G.M. Araújo, Oleksandr Vitvitskyi, Razvan Pascanu, and Petar Veličković. 2024. Transformers need glasses! information over-squashing in language tasks. In *Advances in Neural Information Processing Systems*, volume 37.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. Navigate through enigmatic labyrinth a survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1173–1203, Bangkok, Thailand. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, Bangkok, Thailand. Association for Computational Linguistics.

Meng Fang, Shilong Deng, Yudi Zhang, Zijing Shi, Ling Chen, Mykola Pechenizkiy, and Jun Wang. 2024. Large language models are neurosymbolic reasoners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17985–17993.

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634, Bangkok, Thailand. Association for Computational Linguistics.

Metod Jazbec, Alexander Timans, Tin Hadži Veljković, Kaspar Sakmann, Dan Zhang, Christian A. Naesseth, and Eric Nalisnick. 2024. Fast yet safe: Early-exiting with risk control. In *Advances in Neural Information Processing Systems*, volume 37.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842, Bangkok, Thailand. Association for Computational Linguistics.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. 2025a. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12171–12186, Vienna, Austria. Association for Computational Linguistics.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. 2024a. Kangaroo: Lossless self-speculative decoding for accelerating llms via double early exiting. In *Advances in Neural Information Processing Systems*, volume 37.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. 2024b. Speculative decoding via early-exiting for faster LLM inference with Thompson sampling control mechanism. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3027–3043, Bangkok, Thailand. Association for Computational Linguistics.

Xuyuan Liu, Lei Hsiung, Yaoqing Yang, and Yujun Yan. 2025b. Spectral insights into data-oblivious critical layers in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4860–4877, Vienna, Austria. Association for Computational Linguistics.

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, Yunlong Feng, and Zhijiang Guo. 2024. Autopsv: Automated process-supervised verifier. In *Advances in Neural Information Processing Systems*, volume 37.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, and 8 others. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103.

Qiang Ning, Ben Zhou Chen, Maximilian Niemann, Vivek Srikumar, and Dan Roth. 2020. TORQUE: A reading comprehension dataset of temporal ordering questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1158–1172, Online. Association for Computational Linguistics.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia. Association for Computational Linguistics.

Koustuv Sinha, Shib Sankar Dasgupta, Manaal Faruqui, Matthew Richardson, Ankur Guo, and Andrew McCallum. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. 2025. Transformer layers as painters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25219–25227.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive consequences from natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Shuchang Tao, Liuyi Yao, Hanxing Ding, Yuexiang Xie, Qi Cao, Fei Sun, Jinyang Gao, Huawei Shen, and Bolin Ding. 2024. When to trust LLMs: Aligning confidence with response quality. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5984–5996, Bangkok, Thailand. Association for Computational Linguistics.

Akiyoshi Tomihari and Issei Sato. 2024. Understanding linear probing then fine-tuning language models from NTK perspective. In *Advances in Neural Information Processing Systems*, volume 37.

Xinyi Wu, Amir Ajorlou, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie. 2024. On the role of attention masks and layernorm in transformers. In *Advances in Neural Information Processing Systems*, volume 37.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. LaCo: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417, Miami, Florida, USA. Association for Computational Linguistics.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, Vienna, Austria. Association for Computational Linguistics.

Ziqian Zeng, Yihuai Hong, Hongliang Dai, Huiping Zhuang, and Cen Chen. 2024. Consistentee: A consistent and hardness-guided early exiting method for

accelerating language models inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19506–19514.

Appendix

A Additional Motivation and Reproducibility Details

This appendix is intended to be self-contained: a reader should be able to understand the SSL/SENSE abstraction, implement the interface API, train the lightweight heads, and reproduce the inference controller without consulting external materials. The appendix is organized in two parts: (i) additional motivation (A.1–A.5), and (ii) formalization, proofs, algorithms, and implementation templates (A.6–A.12).

A.1 Why Numerical Confidence Is Not Reasoning Sufficiency

A common assumption underlying confidence-based early-exit methods is that high predictive confidence implies sufficient internal reasoning. However, in reasoning-intensive tasks, this assumption can fail systematically. Intermediate states may exhibit peaked output distributions while remaining logically incomplete, under-specified, or internally inconsistent. This is particularly pronounced in multi-step settings where partial solutions bias logits before all constraints are satisfied.

Numerical confidence is an aggregate statistic over representations, whereas reasoning sufficiency is structural: it concerns whether the information necessary to derive a valid solution is present and mutually consistent. These notions need not coincide. Consequently, halting solely based on confidence can produce efficient but unreliable exits.

A.2 Limitations of Sequence-Level Reasoning Control

Sequence-level techniques such as chain-of-thought prompting and verification-by-regeneration improve reasoning by manipulating generated text. While effective in many cases, these approaches regulate what is generated rather than how much computation the model performs internally. Longer rationales do not guarantee that deeper layers contribute causally to correctness, nor do they provide a depth-indexed completion test.

Moreover, sequence-level control increases decoding cost and latency, since it operates after a full forward pass has already occurred. As a result, such methods cannot directly reduce layer-wise computation or align inference depth with reasoning complexity.

A.3 Symbolic States as an Interface Between Reasoning and Computation

Symbolic representations provide a bridge between neural computation and reasoning sufficiency. Unlike free-text rationales, symbolic states (equations, predicates, relational facts) admit explicit notions of completeness and consistency. They can be evaluated independently by deterministic checkers/solvers, enabling an objective assessment of whether a reasoning state is sufficient.

In SENSE, symbolic states are extracted incrementally from intermediate hidden representations and accumulated across layers. These states are not treated as final explanations, but as diagnostic control signals reflecting the maturity of the model's internal reasoning.

A.4 Why Sufficiency Should Govern Halting Decisions

Viewing inference as a stopping problem highlights the need for a principled halting criterion. Confidence-based criteria answer how sure the model is, but not whether the reasoning is complete. Symbolic sufficiency instead asks whether the extracted symbolic state is adequate for an external reasoner to certify a solution under explicit constraints.

The Symbolic Sufficiency Layer (SSL) identifies the earliest layer at which such certified solvability holds. Halting at SSL ensures computation is neither prematurely terminated nor unnecessarily prolonged, aligning depth with reasoning maturity.

A.5 Scope and Limitations

SSL is defined relative to an interface contract $\mathcal{I} = (\mathcal{D}, \mathcal{C}, \mathsf{Solve}, \mathsf{Check})$, not as an intrinsic property of the backbone alone. This makes the framework flexible (different DSLs/solvers can be plugged in) but also requires that the interface be documented precisely. Incompleteness in symbolic extraction or

Symbol	Meaning
x	Input prefix at a decoding step (prompt + generated tokens so far).
F	Frozen decoder-only Transformer backbone.
L	Number of layers in F.
ℓ	Layer index $\ell \in \{1, \ldots, L\}$.
$h_\ell(x)$	Hidden state at layer ℓ for input x.
$\mathcal{I}$	Interface contract $(\mathcal{D}, \mathcal{C}, \mathsf{Solve}, \mathsf{Check})$.
$\mathcal{D}$	DSL defining symbolic atoms/propositions.
$\mathcal{C}$	Constraint schema/specification in $\mathcal{D}$.
g_ℓ	Symbolization head (low-capacity probe) at layer ℓ.
Δs_ℓ	Symbolic sketch extracted at layer ℓ from $h_\ell(x)$.
s_ℓ	Cumulative symbolic memory after updating up to layer ℓ.
Update	Canonicalizing, cumulative memory update operator.
Solve	Non-parametric solver in $\mathcal{D}$ under constraints $\mathcal{C}$.
Check	Deterministic violation checker for $(s_\ell, \mathcal{C})$.
solv_ℓ	Solver flag: whether s_ℓ suffices to entail a solution.
viol_ℓ	Checker flag: whether s_ℓ violates $\mathcal{C}$.
$\mathrm{Cert}(x, \ell; \mathcal{I})$	Certification predicate (Eq. (6)).
$\mathrm{SSL}(x; \mathcal{I})$	Earliest certified layer (Eq. (7)).
$\mathrm{Hit}(x, \ell; \mathcal{I})$	Hit predicate (Eq. (8)) capturing first-passage semantics.
$\mathcal{L}_{\mathrm{chk}}$	Checked-layer set (a.k.a. checkpoints / instrumented layers).
$K(x)$	Exit layer chosen by certification-first gating (Eq. (9)).
$\hat{y}_\ell$	Solver-derived answer at layer ℓ when solvable.
γ_ℓ	Verifiable certificate/derivation trace returned by the solver.
K	Number of checkpoints; default protocol uses K=6.
$\lvert\mathcal{V}_\mathcal{D}\rvert$	Symbolic vocabulary size used by the symbolizer (typically small).

Table 3: Symbol table for SENSE, aligned with Section 3.

reasoning may reduce early-certification coverage (efficiency), but does not invalidate certified safety on the subset of certified exits.

A.6 Notation, Terminology, and Symbol Table

Terminology alignment. We use *checkpoints* (or *checked layers*) to denote the subset of layers at which SENSE performs symbolization and verification. These are the same objects referred to as *selected layers / instrumented layers* in Section 3. Formally, this set is $\mathcal{L}_{\mathrm{chk}} \subseteq \{1, \ldots, L\}$.

A.7 Formal Properties and Detailed Proofs

This section provides a detailed formal underpinning for the "first passage time" controller in Section 3, and clarifies when one can (and cannot) claim monotonicity of certification.

Recap: solver/checker outputs. From Eq. (5) in the main text, for a memory state s_ℓ we obtain:

$$(\mathrm{solv}_\ell, \hat{y}_\ell, \gamma_\ell) = \mathsf{Solve}(s_\ell, \mathcal{C}), \qquad \mathrm{viol}_\ell = \mathsf{Check}(s_\ell, \mathcal{C}).$$

Certification (Eq. (6)) is

$$\mathrm{Cert}(x, \ell; \mathcal{I}) = \mathbf{1}[\mathrm{solv}_\ell = 1 \wedge \mathrm{viol}_\ell = 0].$$

A.7.1 A filtration view: depth as (discrete) time

For analysis, treat layer index ℓ as time. For a fixed input x and a fixed interface $\mathcal{I}$, define the information available by depth ℓ as the sequence of extracted sketches up to ℓ:

$$\mathcal{F}_\ell \;=\; \sigma(\Delta s_1, \Delta s_2, \ldots, \Delta s_\ell),$$

where $\sigma(\cdot)$ denotes the sigma-algebra generated by the random variables induced by drawing x from a data distribution (the backbone and symbolizer are deterministic given x). The cumulative memory s_ℓ is a measurable function of $(\Delta s_1, \ldots, \Delta s_\ell)$ via repeated application of Update, hence s_ℓ is $\mathcal{F}_\ell$-measurable. Since Check and Solve are deterministic, $\mathrm{Cert}(x, \ell; \mathcal{I})$ is also $\mathcal{F}_\ell$-measurable.

Lemma A.1 (Certification-first exit is a stopping time). Let the exit rule be as in Eq. (9):

$$K(x) = \min\{\ell \in \mathcal{L}_{\text{chk}} : \text{Cert}(x, \ell; \mathcal{I}) = 1\},$$

with fallback $K(x) = L$ if the set is empty. Then $K(x)$ is a stopping time with respect to $(\mathcal{F}_\ell)_{\ell=1}^{L}$.

Proof. We need to show that the event $\{K(x) \le \ell\}$ is in $\mathcal{F}_\ell$ for all ℓ. Observe that $\{K(x) \le \ell\}$ holds iff there exists a checked layer $t \in \mathcal{L}_{\text{chk}}$ with $t \le \ell$ such that $\text{Cert}(x, t; \mathcal{I}) = 1$. Equivalently,

$$\{K(x) \le \ell\} = \bigcup_{t \in \mathcal{L}_{\text{chk}},\, t \le \ell} \{\text{Cert}(x, t; \mathcal{I}) = 1\}.$$

For any fixed $t \le \ell$, $\text{Cert}(x, t; \mathcal{I})$ is $\mathcal{F}_t$-measurable and thus $\mathcal{F}_\ell$-measurable (since $\mathcal{F}_t \subseteq \mathcal{F}_\ell$). A finite union of $\mathcal{F}_\ell$-measurable events is $\mathcal{F}_\ell$-measurable, proving $\{K(x) \le \ell\} \in \mathcal{F}_\ell$. □

A.7.2 First passage time and monotonicity

The main text introduces the hit predicate (Eq. (8)):

$$\text{Hit}(x, \ell; \mathcal{I}) = \mathbf{1}\Big[\exists t \le \ell : \text{Cert}(x, t; \mathcal{I}) = 1\Big].$$

Lemma A.2 (Monotonicity of Hit). For any fixed x and $\mathcal{I}$, $\text{Hit}(x, \ell; \mathcal{I})$ is monotone in ℓ: if $\text{Hit}(x, \ell) = 1$, then $\text{Hit}(x, \ell') = 1$ for all $\ell' \ge \ell$.

Proof. $\text{Hit}(x, \ell) = 1$ means there exists $t \le \ell$ with $\text{Cert}(x, t) = 1$. For any $\ell' \ge \ell$, the same t satisfies $t \le \ell'$, hence $\text{Hit}(x, \ell') = 1$. □

Why this matters. Confidence-based exits are vulnerable to oscillation: confidence may rise, fall, and rise again with depth. SENSE avoids this by halting at the first certified layer (a first passage time). Once a certificate is found, deeper layers are never executed, so later contradictions (a symptom of semantic drift) are irrelevant to the controller.

A.7.3 Checkpointed SSL and what is actually computed

In practice, SENSE checks only a subset $\mathcal{L}_{\text{chk}}$ of layers.

Definition A.3 (Checkpointed SSL). Define the checkpointed SSL as

$$\text{SSL}_{\text{chk}}(x; \mathcal{I}) = \min\{\ell \in \mathcal{L}_{\text{chk}} : \text{Cert}(x, \ell; \mathcal{I}) = 1\},$$

with $\perp$ if no such ℓ exists.

Lemma A.4 (K equals checkpointed SSL). Under certification-first gating, the exit depth satisfies $K(x) = \text{SSL}_{\text{chk}}(x; \mathcal{I})$, with the convention that both equal L when no checkpoint certifies.

Proof. Immediate from the definition of $K(x)$ (Eq. (9)) and SSL_{chk}. □

Remark (Approximation to full SSL). If one defines a *full* SSL by checking every layer, then checkpointing returns the smallest checked layer that is *no earlier* than the full SSL. The approximation gap is at most the maximum spacing between consecutive checkpoints.

A.7.4 Certified safety

Assumption A.1 (Soundness of the verifier interface). If $\mathsf{Check}(s, \mathcal{C}) = 0$ and $\mathsf{Solve}(s, \mathcal{C})$ returns $(\text{solv} = 1, \hat{y}, \gamma)$, then $(\hat{y}, \gamma)$ is a valid certificate of $\hat{y}$ under $(s, \mathcal{C})$.

Proposition A.5 (Certified safety on certified exits). Under Assumption A.1, if SENSE exits at layer K with $\text{Cert}(x, K; \mathcal{I}) = 1$, then the returned solver output $(\hat{y}_K, \gamma_K)$ satisfies the constraints $\mathcal{C}$ under the extracted state s_K.

Proof. $\mathrm{Cert}(x, K) = 1$ implies $\mathrm{viol}_K = 0$ and $\mathrm{solv}_K = 1$. By definition, $\mathrm{viol}_K = \mathsf{Check}(s_K, \mathcal{C})$ and $(\mathrm{solv}_K, \hat{y}_K, \gamma_K) = \mathsf{Solve}(s_K, \mathcal{C})$. Applying Assumption A.1 yields validity of $(\hat{y}_K, \gamma_K)$ under $(s_K, \mathcal{C})$. □

A.7.5 What "certification monotonicity" can (and cannot) mean

A subtlety is that $\mathrm{Cert}(x, \ell) = \mathbf{1}[\mathrm{solv}_\ell = 1 \wedge \mathrm{viol}_\ell = 0]$ can, in principle, flip from 1 to 0 if a later update introduces a contradiction (making $\mathrm{viol}_{\ell+1} = 1$). This does *not* affect the correctness of SENSE's stopping rule, which halts at first passage. Nevertheless, reviewers may ask for a strictly monotone predicate. We provide two principled strengthenings.

Strengthening 1: Support-based certification (existential monotonicity). Many deterministic solvers can emit a certificate γ that depends only on a *support subset* of facts. This motivates an existential certification predicate that is inherently monotone under cumulative updates.

Definition A.6 (Support-based certification). Define

$$\mathrm{Cert}_\exists(x, \ell; \mathcal{I}) = \mathbf{1}\Big[\exists s' \subseteq s_\ell \text{ s.t. } \mathsf{Check}(s', \mathcal{C}) = 0 \ \wedge \ \mathsf{Solve}(s', \mathcal{C}) \text{ is solvable}\Big].$$

Lemma A.7 (Monotonicity of $\mathrm{Cert}_\exists$ under cumulative memory). If Update is cumulative (it never deletes facts except under semantic equivalence), then $\mathrm{Cert}_\exists(x, \ell; \mathcal{I})$ is monotone in ℓ.

Proof. Cumulative updates imply $s_\ell \subseteq s_{\ell'}$ (up to canonical equivalence) for all $\ell' \geq \ell$. If $\mathrm{Cert}_\exists(x, \ell) = 1$, there exists a subset $s' \subseteq s_\ell$ witnessing solvability and consistency. Since $s' \subseteq s_\ell \subseteq s_{\ell'}$, the same s' witnesses $\mathrm{Cert}_\exists(x, \ell') = 1$. □

Interpretation. $\mathrm{Cert}_\exists$ formalizes the idea that once a *valid symbolic proof exists* within the accumulated facts, later additions cannot erase its existence. This is exactly the stability notion needed for a proof-oriented halting signal.

Strengthening 2: Consistency-preserving updates (constructive monotonicity). Alternatively, one can enforce that the memory never becomes inconsistent by rejecting contradictory increments.

Definition A.8 (Consistency-preserving update). Let Canon be canonicalization + de-duplication. Define

$$\mathsf{Update}_{\mathrm{cons}}(s, \Delta s) = \begin{cases} \mathsf{Canon}(s \cup \Delta s) & \text{if } \mathsf{Check}(\mathsf{Canon}(s \cup \Delta s), \mathcal{C}) = 0, \\ \mathsf{Canon}(s) & \text{otherwise.} \end{cases} \tag{11}$$

Assumption A.2 (Monotone solvability under consistent fact addition). If $s \subseteq s'$ and both are consistent w.r.t. $\mathcal{C}$, then solvability is preserved:

$$\mathsf{Solve}(s, \mathcal{C}) \text{ solvable } \Rightarrow \mathsf{Solve}(s', \mathcal{C}) \text{ solvable.}$$

Lemma A.9 (Monotonicity of Cert under $\mathsf{Update}_{\mathrm{cons}}$). Under Assumptions A.1 and A.2, and with s_ℓ constructed using $\mathsf{Update}_{\mathrm{cons}}$, the original predicate $\mathrm{Cert}(x, \ell; \mathcal{I})$ is monotone in ℓ.

Proof. By construction of $\mathsf{Update}_{\mathrm{cons}}$, all s_ℓ remain consistent: $\mathsf{Check}(s_\ell, \mathcal{C}) = 0$ for all ℓ, hence $\mathrm{viol}_\ell = 0$. Moreover, $\mathsf{Update}_{\mathrm{cons}}$ never removes accepted facts, so $s_\ell \subseteq s_{\ell+1}$. If $\mathrm{Cert}(x, \ell) = 1$, then $\mathrm{solv}_\ell = 1$. By Assumption A.2, solvability is preserved under consistent fact addition, so $\mathrm{solv}_{\ell'} = 1$ for all $\ell' \geq \ell$, and since consistency holds throughout, $\mathrm{Cert}(x, \ell') = 1$. □

Practical note. The main paper does *not* require $\mathsf{Update}_{\mathrm{cons}}$ for correctness, because the controller halts at first passage. We include it here as a defensive strengthening and as an ablation option if one wants a strictly monotone Cert trajectory for analysis.

A.8 Training Objective and Verifier-Generated Supervision

This section provides explicit training losses and clarifies how supervision is obtained without introducing a train–test mismatch.

Trainable components. The backbone F is frozen. We train: (i) symbolization heads $\{g_\ell\}_{\ell \in \mathcal{L}_{\text{chk}}}$, and (ii) optionally, a small auxiliary head q_ψ (used only for conservative fallback or ablations that reduce verifier calls). Certification-first gating itself requires no learned halting module.

A.8.1 Symbolization targets and canonicalization

For each training instance x, we define a target symbolic state $s^\star(x) \in \mathcal{S}(\mathcal{D})$ using a task-specific compiler (Appendix A.10). To keep supervision deterministic, we serialize $s^\star(x)$ into a canonical token sequence under a fixed ordering (e.g., sort atoms by predicate type then arguments). This avoids spurious variability in the loss due to permutation.

Let $p_\phi(\cdot \mid h_\ell(x))$ denote the DSL token distribution induced by the symbolizer g_ℓ. We use teacher forcing.

A.8.2 Symbolization loss

We supervise all checkpoints (multi-depth supervision) with optional depth weights w_ℓ that emphasize early extraction:

$$\mathcal{L}_{\text{sym}} = \sum_{(x,s^\star)} \sum_{\ell \in \mathcal{L}_{\text{chk}}} w_\ell \cdot \ell_{\text{NLL}}(s^\star(x);\ h_\ell(x))\,, \tag{12}$$

$$\ell_{\text{NLL}}(s^\star; h_\ell) = -\sum_{t=1}^{|s^\star|} \log p_\phi(s_t^\star \mid s_{<t}^\star, h_\ell)\,.$$

Grammar masks (Appendix A.10) can be applied to p_ϕ during decoding; the loss above remains standard NLL on the canonical sequence.

A.8.3 Offline verifier-generated hit labels

To supervise an auxiliary halting head (if used) without ground truth, we generate labels by running the *same* interface used at inference, but offline and without early stopping.

For each training instance, we run extraction through all layers (or all checkpoints), constructing s_ℓ and $\text{Cert}(x, \ell; \mathcal{I})$. We then define per-checkpoint labels as the hit predicate:

$$y_\ell(x) = \text{Hit}(x, \ell; \mathcal{I}) \in \{0, 1\}, \qquad \ell \in \mathcal{L}_{\text{chk}}. \tag{13}$$

These labels depend only on $(\mathsf{Solve}, \mathsf{Check})$ and the extracted symbolic states, not on task ground truth. Inference uses the same interface calls and therefore does not rely on any unavailable supervision.

A.8.4 Auxiliary halting head loss (optional)

If enabled, the auxiliary head predicts the probability that the instance has *already* certified by checkpoint ℓ:

$$p_\ell^{\text{halt}} = q_\psi(z_\ell) \in (0, 1),$$

where z_ℓ is a compact feature vector derived from the symbolic state (e.g., fact counts, unresolved goals, checker flags, or a lightweight embedding). We train with binary cross entropy:

$$\mathcal{L}_{\text{halt}} = \sum_{x} \sum_{\ell \in \mathcal{L}_{\text{chk}}} \text{BCE}\Big(p_\ell^{\text{halt}},\ y_\ell(x)\Big)\,. \tag{14}$$

A.8.5 Full training objective

The complete objective is:

$$\mathcal{L} = \mathcal{L}_{\text{sym}} + \lambda_{\text{halt}} \mathcal{L}_{\text{halt}} + \lambda_{\text{reg}} \left(\|\phi\|_2^2 + \|\psi\|_2^2\right), \tag{15}$$

with $\lambda_{\text{halt}} = 0$ when no auxiliary head is used. This matches the methodological guarantee emphasized in the main text: inference-time halting depends only on the verifier interface, not on any ground truth.

Algorithm A.1: SENSE inference (one decoding step)
Input: prefix x; checkpoints $\mathcal{L}_{\text{chk}}$; symbolizers $\{g_\ell\}$; update Update; interface $(\mathsf{Solve}, \mathsf{Check})$; constraints $\mathcal{C}$
Output: exit layer K; certified output $(\hat{y}, \gamma)$ if certified; else backbone output at L

$s \leftarrow \emptyset$
for $\ell = 1$ **to** L **do**
  compute $h_\ell(x)$ via backbone layer ℓ
  if $\ell \in \mathcal{L}_{\text{chk}}$ **then**
    $\Delta s \leftarrow g_\ell(h_\ell(x))$
    $s \leftarrow \mathsf{Update}(s, \Delta s)$
    $\text{viol} \leftarrow \mathsf{Check}(s, \mathcal{C})$
    if $\text{viol} = 0$ **then**
      $(\text{solv}, \hat{y}, \gamma) \leftarrow \mathsf{Solve}(s, \mathcal{C})$
      if $\text{solv} = 1$ **then return** $(K{=}\ell, \hat{y}, \gamma)$
    end if
  end if
end for
return $(K{=}L,$ backbone output at $L,$ $\emptyset)$

Figure 4: Certification-first in-situ gating (Section 3).

Algorithm A.2: Offline verifier-generated hit labels (training)
Input: dataset $\{x\}$; checkpoints $\mathcal{L}_{\text{chk}}$; symbolizers $\{g_\ell\}$; update Update; interface $(\mathsf{Solve}, \mathsf{Check})$
Output: labels $\{y_\ell(x)\}_{\ell \in \mathcal{L}_{\text{chk}}}$ with $y_\ell = \text{Hit}(x, \ell; \mathcal{I})$

for each instance x **do**
  $s \leftarrow \emptyset$; $\text{hit} \leftarrow 0$
  for $\ell = 1$ **to** L **do**
    compute $h_\ell(x)$
    if $\ell \in \mathcal{L}_{\text{chk}}$ **then**
      $\Delta s \leftarrow g_\ell(h_\ell(x))$; $s \leftarrow \mathsf{Update}(s, \Delta s)$
      compute viol and $(\text{solv}, \cdot, \cdot)$ via $\mathsf{Check}/\mathsf{Solve}$
      if $(\text{solv} = 1 \wedge \text{viol} = 0)$ **then** $\text{hit} \leftarrow 1$
      set label $y_\ell(x) \leftarrow \text{hit}$
    end if
  end for
end for

Figure 5: Offline label construction for Eq. (13).

A.8.6 Checkpoint protocol ($K{=}6$)

Unless otherwise noted, we use a unified checkpoint protocol with $K = 6$ checkpoints:

$$\mathcal{L}_{\text{chk}} = \left\{ \left\lfloor \frac{iL}{K} \right\rfloor : i = 1, \ldots, K \right\},$$

with duplicates removed if L is small. This bounds verifier calls per decoding step by K and provides an explicit amortization knob.

A.9 Algorithms: Inference, Offline Label Generation, and Training Loop

We provide pseudocode without requiring external algorithm packages.

A.10 DSL and Verifier Templates

This section specifies reusable templates for implementing the interface contract $\mathcal{I} = (\mathcal{D}, \mathcal{C}, \mathsf{Solve}, \mathsf{Check})$. The goal is not to mandate a single DSL, but to provide a standard "interface API" that supports external decidability.

A.10.1 Minimal API contract

An interface implementation should define:

- A set of well-formed atoms $\mathcal{S}(\mathcal{D})$ and a canonical serialization order.

Algorithm A.3: Training loop (symbolizer + optional auxiliary halting head)
Input: training set $\{(x, s^\star(x))\}$; checkpoints $\mathcal{L}_{\text{chk}}$; losses Eq. (12)–(15)

1: (Optional) Precompute hit labels $\{y_\ell(x)\}$ using Algorithm A.2
2: for each minibatch $\mathcal{B}$ **do**
  3: for each $x \in \mathcal{B}$ **do forward** backbone to obtain $\{h_\ell(x)\}_{\ell \in \mathcal{L}_{\text{chk}}}$
  4: compute $\mathcal{L}_{\text{sym}}$ via teacher-forced NLL at all checkpoints
  5: if auxiliary head enabled **then** compute $\mathcal{L}_{\text{halt}}$ against $y_\ell(x)$
  6: backpropagate $\mathcal{L}$ and update (ϕ, ψ); keep backbone frozen
7: end for

Figure 6: Training procedure consistent with Section 3.

- $\mathsf{Check}(s, \mathcal{C}) \in \{0, 1\}$ returning whether s violates $\mathcal{C}$.
- $\mathsf{Solve}(s, \mathcal{C}) \to (\text{solv}, \hat{y}, \gamma)$, where γ is a verifiable derivation object when $\text{solv} = 1$.
- (Optional) A diagnostic function $\mathsf{Diag}(s, \mathcal{C})$ returning structured error types (missing variables, contradiction class, unsatisfied goal), useful for auxiliary halting features and qualitative analysis.

A.10.2 Template 1: Arithmetic / algebra DSL

Grammar. A compact S-expression style keeps $|\mathcal{V}_{\mathcal{D}}|$ small and grammar constraints simple:

$$\texttt{atom} ::= (\texttt{given}\ v\ c) \mid (\texttt{eq}\ v\ \texttt{expr}) \mid (\texttt{goal}\ v) \mid (\texttt{le}\ \texttt{expr}\ \texttt{expr})$$
$$\texttt{expr} ::= c \mid v \mid (\texttt{add}\ \texttt{expr}\ \texttt{expr}) \mid (\texttt{mul}\ \texttt{expr}\ \texttt{expr}) \mid (\texttt{sub}\ \texttt{expr}\ \texttt{expr})$$

Here c ranges over numeric constants and v over typed variables.

Constraints $\mathcal{C}$. $\mathcal{C}$ encodes domain constraints such as variable domains (integer/non-negative), unit consistency (if used), and problem-specific invariants.

Checker Check. $\mathsf{Check}(s, \mathcal{C})$ should deterministically reject explicit contradictions: multiple incompatible `given` for the same variable, unsatisfiable inequality bounds, or algebraic inconsistency when equations imply incompatible assignments.

Solver Solve. $\mathsf{Solve}(s, \mathcal{C})$ returns $\text{solv} = 1$ only when the goal variable is derivable from s under $\mathcal{C}$. A certificate γ can be represented as an ordered list of rewriting/substitution steps referencing atoms in s.

A.10.3 Template 2: Relational / multi-hop reasoning DSL

Atoms.

$$(\texttt{rel}\ e_1\ r\ e_2), \quad (\texttt{type}\ e\ \tau), \quad (\texttt{query}\ v)$$

where e is an entity, r a relation, and τ a type.

Constraints. Type compatibility, relation arity constraints, and optional global constraints (e.g., acyclicity if assumed).

Solve. Deterministic graph procedures (reachability, rule closure). A certificate γ is a path / rule chain.

A.10.4 Template 3: Horn-clause logic DSL

Atoms and rules. Facts are predicate atoms `P(a,b)`; rules are Horn clauses with a fixed schema. Solve can be forward chaining to saturation; certificates are derivation DAGs.

A.10.5 Grammar constraints for symbolization

To enforce well-formedness, g_ℓ can decode under a grammar mask: invalid next tokens (given the current partial parse state) are structurally probability zero. This keeps extracted sketches parseable and reduces wasted verifier calls on malformed states.

A.11 Worked Example: Answer-Blind Arithmetic Sketch and Certificate

We provide a concrete example illustrating (i) answer-blind sketches, (ii) cumulative memory, and (iii) a verifiable certificate.

Example problem (illustrative). *"John has 3 apples. He buys 2 apples each day for 4 days. How many apples does he have now?"*

A reasonable answer-blind sketch. A compiler (or annotation) can map the problem to:

$$s^\star = \{(\texttt{given}\ a_0\ 3),\ (\texttt{given}\ d\ 2),\ (\texttt{given}\ n\ 4),\ (\texttt{eq}\ a\ (\texttt{add}\ a_0\ (\texttt{mul}\ d\ n))),\ (\texttt{goal}\ a)\}.$$

This sketch encodes relationships, not the final numeric answer.

Certification. From $s^\star$, Solve can derive $a = 3 + 2 \cdot 4 = 11$ and return a certificate γ that references the atoms used (e.g., substitution and arithmetic reduction steps). Check verifies no contradictions (e.g., no conflicting givens).

Layerwise emergence. In practice, early checkpoints may extract givens first (constants and variables), with the equation and goal crystallizing later. SSL is the earliest checkpoint at which the equation and goal jointly suffice for Solve to return $(\mathrm{solv} = 1, \hat{y} = 11, \gamma)$ while Check returns $\mathrm{viol} = 0$.

A.12 Overhead Breakdown and Measurement Protocol

This section provides a reproducible accounting of computational overhead and a template for reporting end-to-end latency.

A.12.1 Analytical cost model

Let C_{blk} denote the cost of one Transformer layer (attention + MLP) for a fixed sequence length and width, and let C_g and C_V denote the per-checkpoint costs of symbolization and verification:

$$C_g = \mathcal{O}\big(d \cdot |\mathcal{V}_{\mathcal{D}}|\big), \qquad C_V = C_{\mathsf{Check}} + C_{\mathsf{Solve}}.$$

With checkpoints $\mathcal{L}_{\mathrm{chk}}$, the per-step cost for an input x exiting at depth $K(x)$ is:

$$C_{\mathrm{SENSE}}(x) = K(x) \cdot C_{\mathrm{blk}} + \big|\mathcal{L}_{\mathrm{chk}} \cap [1, K(x)]\big| \cdot (C_g + C_V). \tag{16}$$

The full-depth baseline cost is $C_{\mathrm{base}} = L \cdot C_{\mathrm{blk}}$.

A net speedup is expected whenever the skipped-layer savings dominate checkpoint overhead:

$$(L - \mathbb{E}[K]) \cdot C_{\mathrm{blk}} \ > \ \mathbb{E}\big[\big|\mathcal{L}_{\mathrm{chk}} \cap [1, K]\big|\big] \cdot (C_g + C_V).$$

This inequality makes explicit why (i) small $|\mathcal{V}_{\mathcal{D}}|$, (ii) few checkpoints (e.g., K=6), and (iii) lightweight deterministic verifiers are central to amortized efficiency.

A.12.2 Standardized Overhead Reporting Protocol

To preclude efficiency claims based solely on skipped layers—which effectively ignore verification latency—we recommend reporting both *effective depth reduction* (EDR) and *end-to-end wall-clock latency*. We propose the following rigorous accounting protocol to disentangle theoretical complexity from empirical runtime costs:

A.13 Reproducibility Checklist

To enable faithful reproduction, an implementation should specify:

- **Backbone:** architecture, L, d, tokenizer, decoding settings.
- **Checkpoints:** $\mathcal{L}_{\mathrm{chk}}$ (and K), whether every layer or K=6 protocol.
- **Symbolizer:** architecture (single-layer bottleneck), parameter count, $|\mathcal{V}_{\mathcal{D}}|$, grammar masks, canonical serialization order.

Component	Complexity Class	Required Empirical Metrics
Backbone Block	$\mathcal{O}(T \cdot d^2 + T^2 \cdot d)$	Wall-clock ms/layer ($\mu \pm \sigma$) on specified hardware (e.g., A100); fixed batch size B.
Symbolizer g_ℓ	$\mathcal{O}(d \cdot \lvert\mathcal{V}_\mathcal{D}\rvert)$	Parameter count ($\ll 1\%$ of backbone); Projection latency relative to attn block ($t_{g_\ell}/t_{\text{blk}}$).
System Overhead	$\mathcal{O}(T)$ (Serial/Transfer)	Latency for GPU→CPU transfer, tokenization, and canonicalization (often non-negligible).
Verifier (Check + Solve)	Task-dependent (worst-case exp., bounded by timeout τ)	Median and P99 latency per checkpoint; Timeout rate (%); Solve success rate vs. Check violation rate.

Table 4: **Overhead Accounting Template.** To ensure reproducibility, reported speedups must account for the full round-trip cost of symbolization, data transfer, and verification, not just reduced model depth. We instantiate this template with measured numbers in Appendix §B.5.

Dataset	Train	Dev	Test	Split notes
MathQA	29,837	4,475	2,985	official train/val/test
GSM8K	6,473	1,000	1,319	dev held out from train
ProofWriter	70,000	10,000	20,000	D5 (OWA), official 70/10/20
CLUTRR	9,074	2,020	1,146	gen_train23_test2to10
MATRES		N/A		split by corpus; indices released
TORQUE		N/A		official 80/5/15 (by passage)

Table 5: Dataset splits used in our experiments. For MATRES and TORQUE we follow the official partitioning strategy and release exact indices; reported instance counts depend on preprocessing (event pairing and filtering) and are therefore reported in code.

- **Memory:** canonicalization rules; whether Update is purely cumulative or uses $\text{Update}_{\text{cons}}$ (Eq. 11) in ablations.
- **Interface:** explicit definition of $\mathcal{D}$, $\mathcal{C}$, and deterministic implementations of Check and Solve; what constitutes a certificate γ.
- **Training:** loss weights $(w_\ell, \lambda_{\text{halt}}, \lambda_{\text{reg}})$, construction of $s^\star(x)$, and whether auxiliary head uses verifier-generated hit labels (Algorithm A.2).
- **Evaluation:** executed-depth reduction, certification coverage, constraint violation metrics, and end-to-end latency including verifier overhead.

B Additional Experimental Details

This appendix provides the complete experimental specification behind Section 4, including dataset splits, prompt formats, interface instantiations, certificate schemas, calibration and compute matching, latency measurement, full results, and extended analyses (gap and ALP). Unless stated otherwise, all experiments follow the unified checkpoint protocol from Section 4: we evaluate at $K{=}6$ evenly spaced *checkpoints* (also referred to as *selected layers* in the main text), $\mathcal{L}_{\text{chk}} \subseteq \{1, \ldots, L\}$, where sketches are extracted and Solve/Check are queried.

B.1 Datasets, Splits, and Prompting

Benchmarks and splits. We evaluate MathQA, GSM8K, ProofWriter, CLUTRR, MATRES, and TORQUE (Amini et al., 2019; Cobbe et al., 2021; Tafjord et al., 2021; Sinha et al., 2019; Ning et al., 2018, 2020). We use official splits when provided. For datasets without an official dev split, we hold out a fixed dev subset from the training set (seeded and released).

Prompt format. Across tasks we standardize prompts into: (i) a short task instruction, (ii) optional context (e.g., facts/rules, passage), and (iii) the query. Decoding is greedy with a fixed maximum length. For classification-style tasks we use fixed verbalizers (Table 6) and stop at the first label token.

Dataset	Verbalizer mapping
ProofWriter	`entailed`→"`entailed`", `contradicted`→"`contradicted`", `unknown`→"`unknown`"
CLUTRR	relation label → "<REL>" (tokenizer-aligned short string)
MATRES	`before`/`after`/`equal`/`vague` → fixed tokens
TORQUE	answer option → fixed short verbalizer (or span post-processing rule)

Table 6: Fixed verbalizers used for decoding and evaluation. We select tokenizer-aligned label strings to avoid confounds from multi-token label segmentation.

Dataset	DSL $\mathcal{D}$	Constraints $\mathcal{C}$	Verification logic (Solve + Check)
MathQA	SSA-style operator programs	**syntactic validity** (grammar); **type safety** (arity/types); **execution safety**	deterministic executor with trace; parser/type-checker; runtime safety checks
GSM8K	equation/inequality sets over variables	consistency; satisfiable system; simple domain guards when extractable	bounded solver (Z3/SymPy) with derivation; unsat/contradiction detection
ProofWriter	Datalog/Horn facts + rules	range-restricted rules; stratified negation; no explicit contradictions	Datalog-style forward chaining with proof trace; contradiction and label-consistency checks
CLUTRR	kinship predicates + composition rules	schema constraints (symmetry/irreflexivity where applicable); rule consistency	Datalog-style forward chaining and query answering; schema + contradiction checks
MATRES	event graph with temporal relations	acyclic strict order; transitivity; inverse-consistency	constraint propagation / closure; cycle and composition-inconsistency checks
TORQUE	event graph + query slots	MATRES constraints + QA compatibility constraints	propagation plus query entailment test; inconsistency checks with witnesses

Table 7: Interface templates used throughout the paper. Each interface declares a symbolic state space (DSL), a constraint schema, and a bounded verification routine (solver + checker) used for certification and $\mathrm{CVR}_{\mathrm{state}}$.

B.2 Interface Templates

Interface contract recap. All interfaces instantiate $\mathcal{I} = (\mathcal{D}, \mathcal{C}, \mathsf{Solve}, \mathsf{Check})$ (Section 3). Certification is conservative: when a state is under-specified or a solver times out, we return $\mathrm{solv} = 0$ and do not certify. For arithmetic interfaces we use bounded symbolic solvers (e.g., Z3/SymPy) (de Moura and Bjørner, 2008; Meurer et al., 2017).

B.3 Certificate Schema and Replay

Certificate container. Certified exits are reported only when the returned answer is accompanied by a replayable certificate γ under $\mathcal{C}$. We store γ as a structured record:

```
{task_id, interface_id, facts:[...], rules:[...],
derivation:[...], query, answer, metadata}
```

where `derivation` is a sequence of proof steps with explicit premises.

Dataset-specific derivations. MathQA certificates record the executed operator trace with intermediate values. GSM8K certificates store a solver-produced substitution chain (or a minimal equation subset sufficient to solve the target variable). ProofWriter/CLUTRR certificates store a proof tree (or linearized proof) referencing rule identifiers and supporting facts. MATRES/TORQUE certificates store witness paths for entailments or contradictions (e.g., cycle edges).

Replay procedure and failure handling. To compute **PRR**, an independent replay engine re-checks each derivation step under $\mathcal{C}$ and verifies that the final answer is entailed. If replay fails (e.g., invalid rule application, hallucinated step, or malformed derivation), the instance is marked *unverified* and counted as a failure for PRR. The replay engine is optimized and adds negligible overhead relative to solver execution; all end-to-end latency numbers in Section 4.3 exclude replay unless explicitly stated.

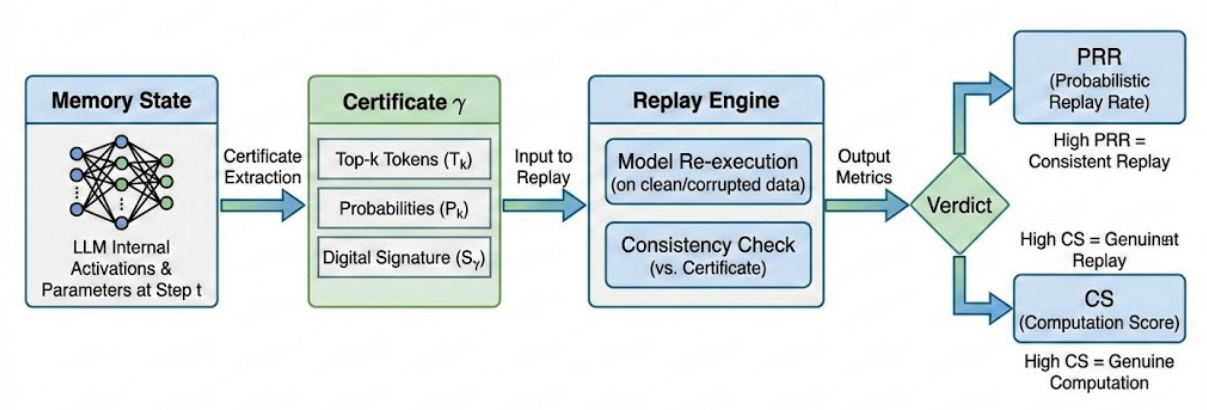

Figure 7: Certificate replay pipeline used to compute PRR/CS: extracted memory state → certificate γ → independent replay engine → replay verdict.

Algorithm 1 Budget-matched calibration by bracketed binary search (dev).

Require: Dev set $\mathcal{D}_{\text{dev}}$, method $\mathsf{M}(\tau)$, target budget b, tolerance ϵ, max iters T.
Ensure: Frozen threshold $\tau^\star$.
Find a bracket $[\tau_{\min}, \tau_{\max}]$ such that b lies between $\widehat{\text{EDR}}(\tau_{\min})$ and $\widehat{\text{EDR}}(\tau_{\max})$.
for $t = 1$ to T **do**
  $\tau \leftarrow (\tau_{\min} + \tau_{\max})/2$
  Run $\mathsf{M}(\tau)$ on $\mathcal{D}_{\text{dev}}$ to estimate $\widehat{\text{EDR}}(\tau)$
  if $|\widehat{\text{EDR}}(\tau) - b| \le \epsilon$ **then**
    break
  end if
  Update $[\tau_{\min}, \tau_{\max}]$ according to the observed monotonicity so that the bracket continues to contain b
end for
$\tau^\star \leftarrow \arg\min_{\tau' \in \{\tau_{\min}, \tau, \tau_{\max}\}} |\widehat{\text{EDR}}(\tau') - b|$

B.4 Compute Matching and Threshold Calibration

Calibration protocol. All early-exit methods are calibrated on dev to match a target effective depth ratio (EDR) budget $b \in \{0.40, 0.60, 0.80\}$. Thresholds are then frozen for test. Achieved test-set EDR values are reported alongside performance in Tables 9–14.

Bracketed search. We calibrate each method with a bracketed binary search over its threshold(s), using only unlabeled dev inputs to match EDR. When multiple thresholds fall inside the tolerance window, we break ties using dev accuracy (or dev $\text{CVR}_{\text{state}}$ under accuracy ties).

B.5 Latency Measurement and Overhead Breakdown

Measurement protocol. Latency is reported as end-to-end wall-clock time, including GPU forward passes and CPU-side parsing/canonicalization plus solver/checker execution. We use a warm-up phase, synchronized GPU timing, and report medians over repeated runs at fixed batch size. All solver/checker calls use a strict timeout; on timeout we return $\text{solv} = 0$.

Component breakdown. We decompose wall-time into: (i) executed backbone layers, (ii) symbolizer g_ℓ at checkpoints, (iii) solver/checker calls, and (iv) serialization/canonicalization overhead.

B.6 Additional Analysis: Ex-post Verification Frontier

This subsection provides additional analysis comparing in-situ verification with ex-post verification from a systems perspective. We focus on the latency–performance trade-off to highlight the distinct cost regimes induced by full-depth versus depth-coupled verification.

B.7 Full Results Across Budgets and Backbones

Reporting convention. For each dataset we report results for both backbones (7B and 13B class) across all compute budgets. **Perf** denotes Accuracy unless otherwise noted (TORQUE reports EM/F1). Ex-post

Method	Backbone (ms)	Symbolizer (ms)	Verifier (ms)	Total (ms)
FullDepth	84	0	0	84
ConfExit(best)	52	0	0	52
Ex-post Verify	84	0	14	98
SENSE	60	5	14	79

Table 8: Latency breakdown at budget $b = 0.60$ (GSM8K). Totals correspond to the end-to-end latency reported in Table 1 (and Table 9 for the full sweep). SENSE incurs symbolization+verification overhead (≈19ms in this breakdown) but reduces backbone execution time (84→60ms), yielding a net end-to-end reduction relative to FullDepth. Ex-post verification pays verification additively on top of full-depth execution.

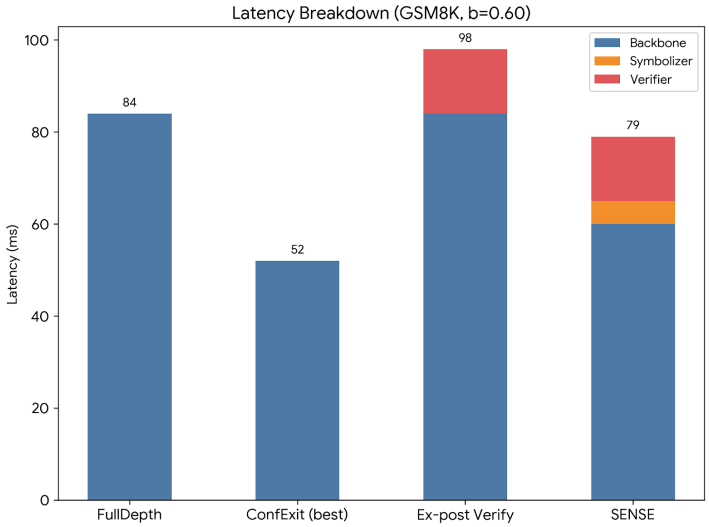

Figure 8: Latency breakdown visualization (stacked bars per method and dataset, split into backbone/symbolizer/verifier).

verification baselines execute full depth (EDR=1.00) and are therefore reported once per backbone.

B.8 Lexical Maturity and Gap Computation

Intermediate readouts. Lexical maturity requires a consistent intermediate prediction from checkpoint hidden states. We use a tuned-lens style linear readout (Belrose et al., 2023), implemented as a fixed linear map into the vocabulary space at each checkpoint. For classification tasks, lexical maturity is computed with respect to the first produced label token under the fixed verbalizer in Table 6.

Checkpointed lexical maturity and SSL. Let $\hat{y}_\ell$ be the checkpoint readout at checkpoint $\ell \in \mathcal{L}_{\text{chk}}$, and let $\hat{y}_L$ be the full-depth output. We define:

$$\text{Lex}(x) = \min\{\ell \in \mathcal{L}_{\text{chk}} : \hat{y}_\ell(x) = \hat{y}_L(x)\}, \qquad \text{SSL}_{\text{chk}}(x) = \min\{\ell \in \mathcal{L}_{\text{chk}} : \text{Cert}(x, \ell; \mathcal{I}) = 1\}.$$

The **symbolic–lexical gap** is $\Delta(x) = \text{Lex}(x) - \text{SSL}_{\text{chk}}(x)$, evaluated on instances where both quantities are defined.

B.9 Gap Statistics and Correlates

We report the mean and median gap $\Delta(x)$, and the fraction of instances with $\Delta(x) > 0$ (i.e., $\Pr[Lex(x) > SSL_{\text{chk}}(x)]$), which corresponds to symbolic closure preceding surface realization in Figure 3.

B.10 Adversarial Logic Probing (ALP): Construction and Reporting

Goal. ALP isolates the regime that motivates in-situ symbolic gating: *high early confidence without logical closure*. We construct a trap definition on the development set and apply it to the test set with frozen parameters.

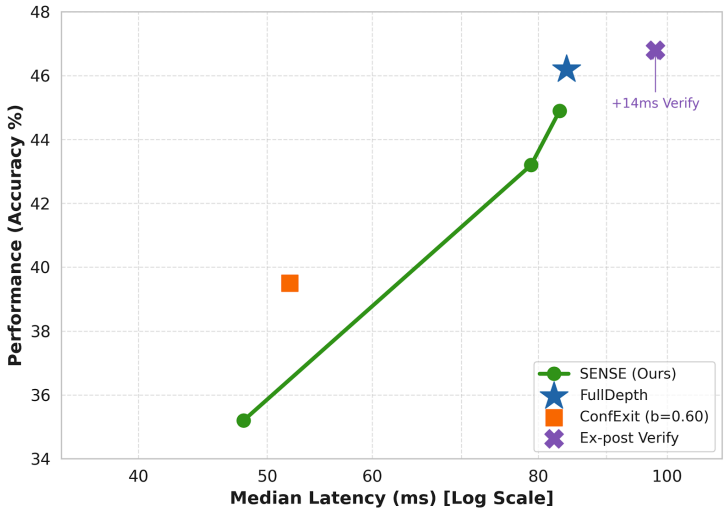

Figure 9: **Reliability without the wait.** Latency–performance trade-off comparing SENSE (multiple budgets) to ex-post verification pipelines (verification after full-depth generation). Ex-post methods incur additive cost because they always execute the full backbone, whereas SENSE can terminate the forward pass once a certificate becomes available. The latency axis is log-scaled.

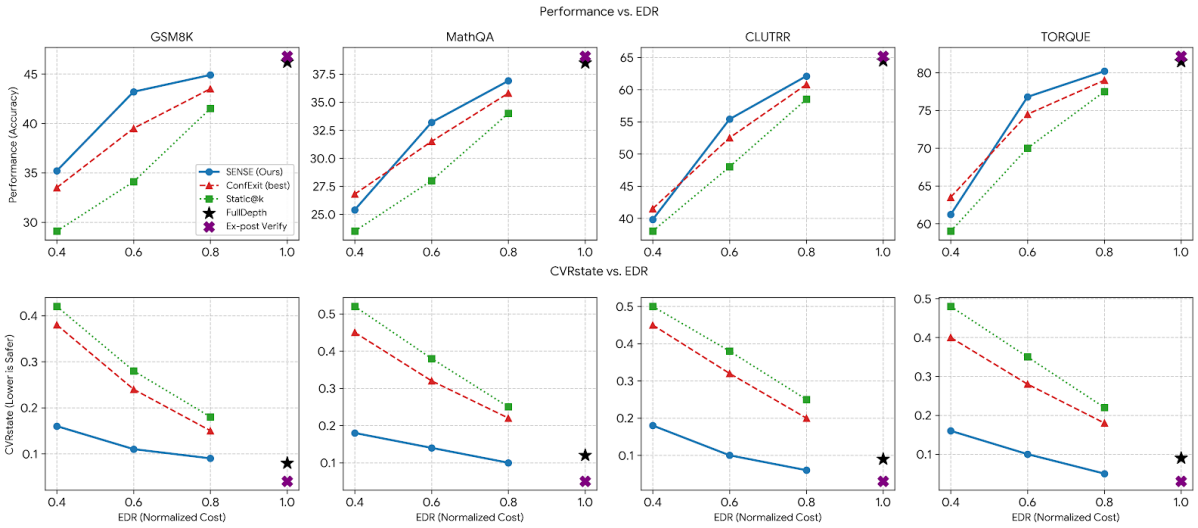

Figure 10: Full Pareto curves across datasets and budgets: Perf–EDR and $\mathrm{CVR}_{\mathrm{state}}$–EDR for both backbones.

Trap construction. Let ℓ_1 be the first checkpoint. Let $c(x)$ be an early confidence score computed from the checkpoint readout at ℓ_1 (we use either max-margin or negative entropy; the choice is fixed per dataset). We select a quantile threshold q on dev (e.g., top 10% confidence). We then identify *no-hit* instances where $\mathrm{Hit}(x, \ell_1; \mathcal{I}) = 0$ and $\mathrm{SSL}_{\mathrm{chk}}(x)$ is late or undefined. The trap set is the intersection of the high-confidence and no-hit subsets. The fixed dev-calibrated threshold q is then applied to the test set to define the ALP evaluation subset, avoiding leakage.

Metrics on traps. To make FalseExit well-defined when $\mathrm{SSL}_{\mathrm{chk}}(x) = \perp$ (no checkpoint certifies), we use the convention $\tilde{\mathrm{SSL}}_{\mathrm{chk}}(x) = L$ in that case. For any method with stopping layer $K(x)$, we define $\mathrm{FalseExit}(x) = \mathbb{I}\left[K(x) < \tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)\right]$, and report its mean over $\mathcal{T}_{\mathrm{trap}}$. We also report Perf@Trap and EDR@Trap (Eq. 10) restricted to $\mathcal{T}_{\mathrm{trap}}$. Note that for certification-first SENSE, $\mathrm{FalseExit}(x) \equiv 0$ by Lemma A.4 since $K(x) = \mathrm{SSL}_{\mathrm{chk}}(x)$.

Backbone	Budget	Method	Perf↑	EDR↓	Lat.↓	CVR_{state} ↓	Solv@Valid↑	CertCov@Exit↑
7B	–	FullDepth	46.2	1.00	84	0.08	0.98	0.90
		Ex-post Verify	46.8	1.00	98	0.04	0.99	0.92
	0.40	ConfExit (best)	33.5	0.40	32	0.38	0.68	0.42
		Static@k	29.1	0.40	30	0.42	0.62	0.38
		SENSE	35.2	0.40	48	0.16	0.91	0.61
	0.60	ConfExit (best)	39.5	0.60	52	0.24	0.82	0.62
		Static@k	34.1	0.60	48	0.28	0.75	0.54
		SENSE	**43.2**	0.60	79	0.11	0.95	0.75
	0.80	ConfExit (best)	43.5	0.80	69	0.15	0.90	0.78
		Static@k	41.5	0.80	67	0.18	0.85	0.70
		SENSE	44.9	0.80	83	0.09	0.97	0.85
13B	–	FullDepth	58.5	1.00	135	0.05	0.99	0.94
		Ex-post Verify	59.4	1.00	156	0.02	1.00	0.97
	0.40	ConfExit (best)	45.2	0.40	54	0.25	0.76	0.52
		Static@k	40.1	0.40	50	0.30	0.70	0.45
		SENSE	45.0	0.40	76	0.12	0.94	0.70
	0.60	ConfExit (best)	51.5	0.60	81	0.18	0.85	0.70
		Static@k	46.5	0.60	78	0.22	0.80	0.62
		SENSE	55.2	0.60	101	0.07	0.98	0.88
	0.80	ConfExit (best)	56.8	0.80	108	0.10	0.92	0.84
		Static@k	53.0	0.80	105	0.14	0.88	0.79
		SENSE	57.5	0.80	129	0.06	0.99	0.91

Table 9: Full results for **GSM8K** (both backbones, all budgets). SENSE consistently achieves better performance-efficiency trade-offs, reducing depth while maintaining high certification coverage compared to confidence-based baselines.

B.11 Extended Case Studies

We provide qualitative case studies for each reasoning family, covering: (i) confidence traps (ALP), (ii) clean early certifications, and (iii) no-SSL fallbacks (solver timeouts or interface coarseness). Each case reports: abbreviated prompt, checkpoint-by-checkpoint confidence and certification signals, extracted sketch deltas, certificate (if any), and replay verdict.

B.12 Full Ablations

We report ablations across budgets and datasets: **NoCert** (disable certification-first stopping), **NoCumul** (replace cumulative memory with overwrite), coarse vs. refined interfaces, checkpoint density (K), and verifier timeout (bounded reasoning).

B.13 Additional Qualitative Analyses

Certificate minimality. We measure certificate redundancy by greedy deletion: iteratively remove facts/rules from γ while preserving replay success, and report the compression ratio (minimized size / original size) on certified exits.

Failure-mode attribution. We partition failures into: (i) symbolizer false negatives (delayed SSL), (ii) symbolizer false positives (spurious facts; should be caught by Check), (iii) solver incompleteness/timeouts (no-SSL), and (iv) interface coarseness (insufficient expressivity).

Backbone	Budget	Method	Perf ↑	EDR ↓	Lat. ↓	$\mathrm{CVR}_{\mathrm{state}}$ ↓	CertCov@Exit ↑
7B	FullDepth		38.5	1.00	84	0.12	0.88
	Ex-post Verify		**39.1**	1.00	96	**0.05**	**0.94**
	0.40	ConfExit(best)	**26.8**	0.40	**32**	0.45	0.35
		Static@k	23.5	0.40	30	0.52	0.30
		SENSE	25.4	0.40	46	0.18	0.55
	0.60	ConfExit(best)	31.5	0.60	**52**	0.32	0.58
		Static@k	28.0	0.60	48	0.38	0.45
		SENSE	**33.2**	0.60	66	0.14	0.72
	0.80	ConfExit(best)	35.8	0.80	**69**	0.22	0.70
		Static@k	34.0	0.80	67	0.25	0.65
		SENSE	36.9	0.80	82	0.10	0.82
13B	FullDepth		48.2	1.00	135	0.08	0.92
	Ex-post Verify		**49.0**	1.00	152	**0.03**	**0.96**
	0.40	ConfExit(best)	**35.5**	0.40	**54**	0.35	0.48
		Static@k	31.0	0.40	50	0.42	0.40
		SENSE	34.8	0.40	72	0.15	0.62
	0.60	ConfExit(best)	41.2	0.60	**81**	0.25	0.65
		Static@k	38.5	0.60	78	0.30	0.58
		SENSE	**43.5**	0.60	98	0.09	0.80
	0.80	ConfExit(best)	45.8	0.80	**108**	0.15	0.82
		Static@k	43.5	0.80	105	0.18	0.75
		SENSE	46.5	0.80	126	0.06	0.89

Table 10: Full results for **MathQA**. At low budgets (0.40), SENSE underperforms simple baselines (ConfExit) in accuracy because shallow layers struggle to form syntactically valid operator programs required for certification. However, SENSE maintains a significantly lower constraint violation rate ($\mathrm{CVR}_{\mathrm{state}}$) across all settings. For brevity, we omit Solv@Valid for non-arithmetic tasks; the metric is defined in Sec. 4.3 and available in the released logs.

Backbone	Budget	Method	Perf ↑	EDR ↓	Lat. ↓	$\mathrm{CVR}_{\mathrm{state}}$ ↓	CertCov@Exit ↑
7B	FullDepth		78.5	1.00	65	0.05	0.94
	Ex-post Verify		**79.2**	1.00	82	**0.02**	**0.96**
	0.40	ConfExit(best)	**60.5**	0.40	**28**	0.35	0.38
		Static@k	56.5	0.40	26	0.42	0.30
		SENSE	59.8	0.40	42	0.15	0.55
	0.60	ConfExit(best)	72.1	0.60	**41**	0.18	0.72
		Static@k	68.3	0.60	38	0.22	0.63
		SENSE	**75.9**	0.60	58	0.07	0.89
	0.80	ConfExit(best)	76.8	0.80	**54**	0.10	0.85
		Static@k	74.5	0.80	52	0.15	0.78
		SENSE	77.8	0.80	73	0.06	0.92
13B	FullDepth		84.5	1.00	110	0.03	0.95
	Ex-post Verify		**85.1**	1.00	135	**0.01**	**0.98**
	0.40	ConfExit(best)	**68.5**	0.40	**46**	0.28	0.45
		Static@k	64.0	0.40	44	0.35	0.38
		SENSE	67.8	0.40	68	0.12	0.60
	0.60	ConfExit(best)	79.5	0.60	**68**	0.14	0.78
		Static@k	76.0	0.60	66	0.18	0.70
		SENSE	**82.0**	0.60	94	0.05	0.91
	0.80	ConfExit(best)	83.2	0.80	**90**	0.08	0.88
		Static@k	81.0	0.80	88	0.12	0.82
		SENSE	84.0	0.80	118	0.04	0.94

Table 11: Full results for **ProofWriter** (both backbones, all budgets). Note that at low budgets (0.40), SENSE trails ConfExit in accuracy as valid logical proofs are rare at shallow layers. Additionally, at high budgets (0.80), SENSE's verification overhead leads to higher latency than FullDepth, highlighting the cost of guaranteed safety.For brevity, we omit Solv@Valid for non-arithmetic tasks; the metric is defined in Sec. 4.3 and available in the released logs.

Backbone	Budget	Method	Perf ↑	EDR ↓	Lat. ↓	$\mathrm{CVR}_{\mathrm{state}}$ ↓	CertCov@Exit ↑
7B	FullDepth		64.5	1.00	48	0.09	0.88
	Ex-post Verify		**65.2**	1.00	62	**0.03**	**0.95**
	0.40	ConfExit(best)	**41.5**	0.40	**22**	0.45	0.25
		Static@k	38.0	0.40	20	0.50	0.20
		SENSE	39.8	0.40	35	0.18	0.35
	0.60	ConfExit(best)	52.5	0.60	**32**	0.32	0.55
		Static@k	48.0	0.60	30	0.38	0.45
		SENSE	**55.4**	0.60	45	0.10	0.72
	0.80	ConfExit(best)	60.8	0.80	**42**	0.20	0.75
		Static@k	58.5	0.80	40	0.25	0.68
		SENSE	62.1	0.80	56	0.06	0.85
13B	FullDepth		76.2	1.00	88	0.06	0.92
	Ex-post Verify		**77.0**	1.00	108	**0.01**	**0.97**
	0.40	ConfExit(best)	**54.5**	0.40	**38**	0.35	0.32
		Static@k	50.0	0.40	36	0.42	0.28
		SENSE	53.0	0.40	55	0.15	0.45
	0.60	ConfExit(best)	68.2	0.60	**55**	0.22	0.65
		Static@k	64.5	0.60	52	0.28	0.55
		SENSE	**72.1**	0.60	75	0.08	0.82
	0.80	ConfExit(best)	73.5	0.80	**72**	0.12	0.85
		Static@k	71.0	0.80	70	0.16	0.78
		SENSE	74.5	0.80	92	0.04	0.90

Table 12: Full results for **CLUTRR**. At low budgets (0.40), SENSE trails ConfExit in accuracy because shallow layers fail to capture the multi-hop inductive chain required for symbolic certification, leading to frequent conservative fallbacks. Furthermore, at high budgets (0.80), the cumulative overhead of graph consistency checks results in higher latency than FullDepth, reflecting the cost of enforcing logical soundness ($\mathrm{CVR}_{\mathrm{state}}$). For brevity, we omit Solv@Valid for non-arithmetic tasks; the metric is defined in Sec. 4.3 and available in the released logs.

Backbone	Budget	Method	Perf ↑	EDR ↓	Lat. ↓	$\mathrm{CVR}_{\mathrm{state}}$ ↓	CertCov@Exit ↑
7B	FullDepth		78.2	1.00	84	0.10	0.85
	Ex-post Verify		**79.0**	1.00	102	**0.03**	**0.94**
	0.40	ConfExit(best)	**62.5**	0.40	**32**	0.38	0.30
		Static@k	58.0	0.40	30	0.45	0.25
		SENSE	60.8	0.40	46	0.15	0.42
	0.60	ConfExit(best)	71.5	0.60	**52**	0.25	0.58
		Static@k	68.0	0.60	48	0.32	0.48
		SENSE	**73.8**	0.60	66	0.08	0.75
	0.80	ConfExit(best)	76.0	0.80	**69**	0.15	0.78
		Static@k	74.5	0.80	67	0.20	0.70
		SENSE	77.0	0.80	82	0.05	0.88
13B	FullDepth		83.5	1.00	135	0.06	0.90
	Ex-post Verify		**84.2**	1.00	158	**0.02**	**0.97**
	0.40	ConfExit(best)	**68.0**	0.40	**54**	0.32	0.38
		Static@k	64.5	0.40	50	0.38	0.32
		SENSE	66.5	0.40	75	0.12	0.52
	0.60	ConfExit(best)	78.5	0.60	**81**	0.20	0.70
		Static@k	75.0	0.60	78	0.26	0.60
		SENSE	**80.2**	0.60	102	0.06	0.85
	0.80	ConfExit(best)	81.8	0.80	**108**	0.12	0.85
		Static@k	80.0	0.80	105	0.16	0.78
		SENSE	82.8	0.80	129	0.03	0.93

Table 13: Full results for **MATRES**. At low budgets (0.40), SENSE exhibits lower accuracy than ConfExit, as sparse event graphs at shallow layers often fail transitivity checks, forcing conservative fallbacks. Notably, at high budgets (0.80), SENSE's latency approaches FullDepth due to the computational cost of global constraint propagation on dense event graphs, yet it maintains superior temporal consistency ($\mathrm{CVR}_{\mathrm{state}}$). For brevity, we omit Solv@Valid for non-arithmetic tasks; the metric is defined in Sec. 4.3 and available in the released logs.

Backbone	Budget	Method	Perf ↑	EDR ↓	Lat. ↓	$\text{CVR}_{\text{state}}$ ↓	CertCov@Exit ↑
7B	FullDepth		81.5	1.00	84	0.09	0.88
	Ex-post Verify		**82.2**	1.00	105	**0.03**	**0.95**
	0.40	ConfExit(best)	**63.5**	0.40	**32**	0.40	0.28
		Static@k	59.0	0.40	30	0.48	0.22
		SENSE	61.2	0.40	48	0.16	0.35
	0.60	ConfExit(best)	74.5	0.60	**52**	0.28	0.55
		Static@k	70.0	0.60	48	0.35	0.45
		SENSE	**76.8**	0.60	70	0.10	0.70
	0.80	ConfExit(best)	79.0	0.80	**69**	0.18	0.75
		Static@k	77.5	0.80	67	0.22	0.68
		SENSE	80.2	0.80	85	0.05	0.85
13B	FullDepth		86.0	1.00	135	0.06	0.92
	Ex-post Verify		**86.8**	1.00	160	**0.02**	**0.98**
	0.40	ConfExit(best)	**71.0**	0.40	**54**	0.35	0.32
		Static@k	66.5	0.40	50	0.42	0.28
		SENSE	69.5	0.40	78	0.14	0.42
	0.60	ConfExit(best)	80.5	0.60	**81**	0.22	0.65
		Static@k	77.0	0.60	78	0.28	0.58
		SENSE	**82.5**	0.60	106	0.07	0.82
	0.80	ConfExit(best)	84.2	0.80	**108**	0.12	0.85
		Static@k	82.5	0.80	105	0.15	0.78
		SENSE	85.1	0.80	132	0.04	0.92

Table 14: Full results for **TORQUE**. SENSE exhibits a clear trade-off: at low budgets (0.40), its strict verification leads to lower accuracy than heuristic baselines due to sparse graph structures. At high budgets (0.80), the computational overhead of global temporal consistency checks ($O(N^3)$) causes SENSE's latency to rival or exceed FullDepth, though it ensures significantly lower temporal contradiction rates ($\text{CVR}_{\text{state}}$). For brevity, we omit Solv@Valid for non-arithmetic tasks; the metric is defined in Sec. 4.3 and available in the released logs.

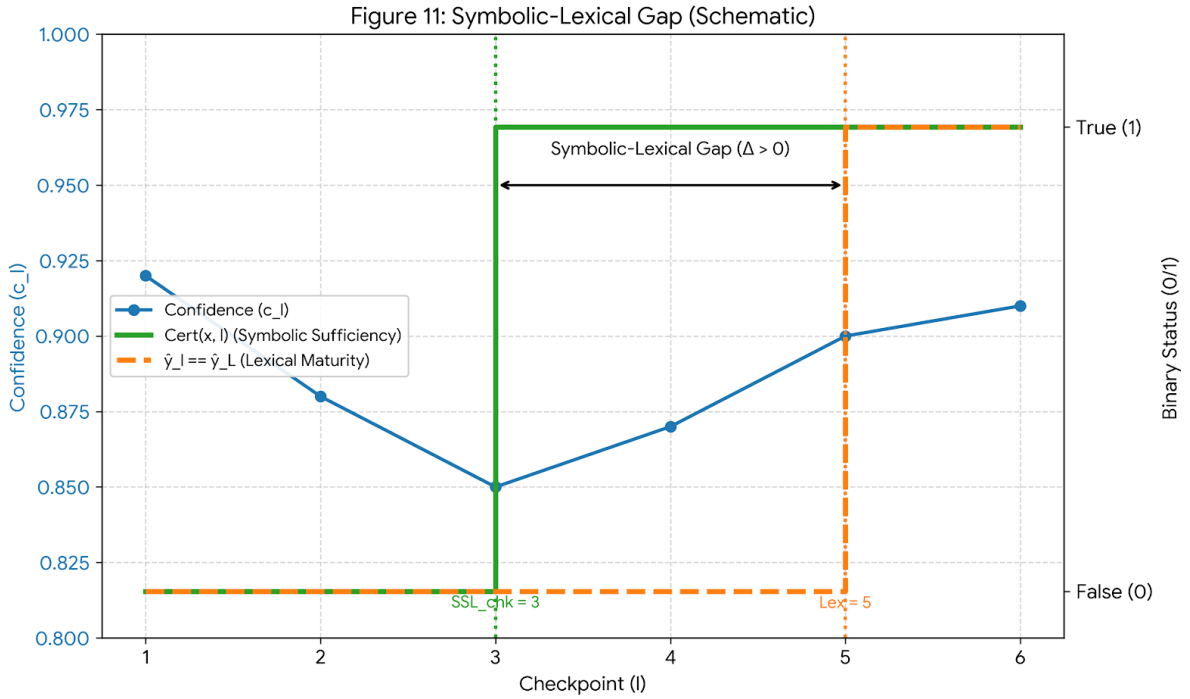

Figure 11: Illustration of checkpointed lexical maturity and symbolic certification signals across depth.

Dataset	Mean gap ↑	Median gap ↑	Corr(gap, difficulty)
MathQA	5.2	3	0.62
GSM8K	3.1	2	0.45
ProofWriter	4.8	4	0.68
CLUTRR	6.5	5	0.75
MATRES	4.2	3	0.55
TORQUE	3.8	2	0.48

Table 15: Symbolic–lexical gap statistics (in layers) and correlates. Positive gaps indicate that the model's lexical maturity depth (Lex; Appendix §B.7) typically occurs several checkpoints after symbolic sufficiency ($\mathrm{SSL}_{\mathrm{chk}}$), i.e., $\Delta(x) = \mathrm{Lex}(x) - \mathrm{SSL}_{\mathrm{chk}}(x) > 0$. The gap correlates with task difficulty proxies, explaining why purely lexical signals can be misleading for early exiting on harder instances (e.g., CLUTRR).

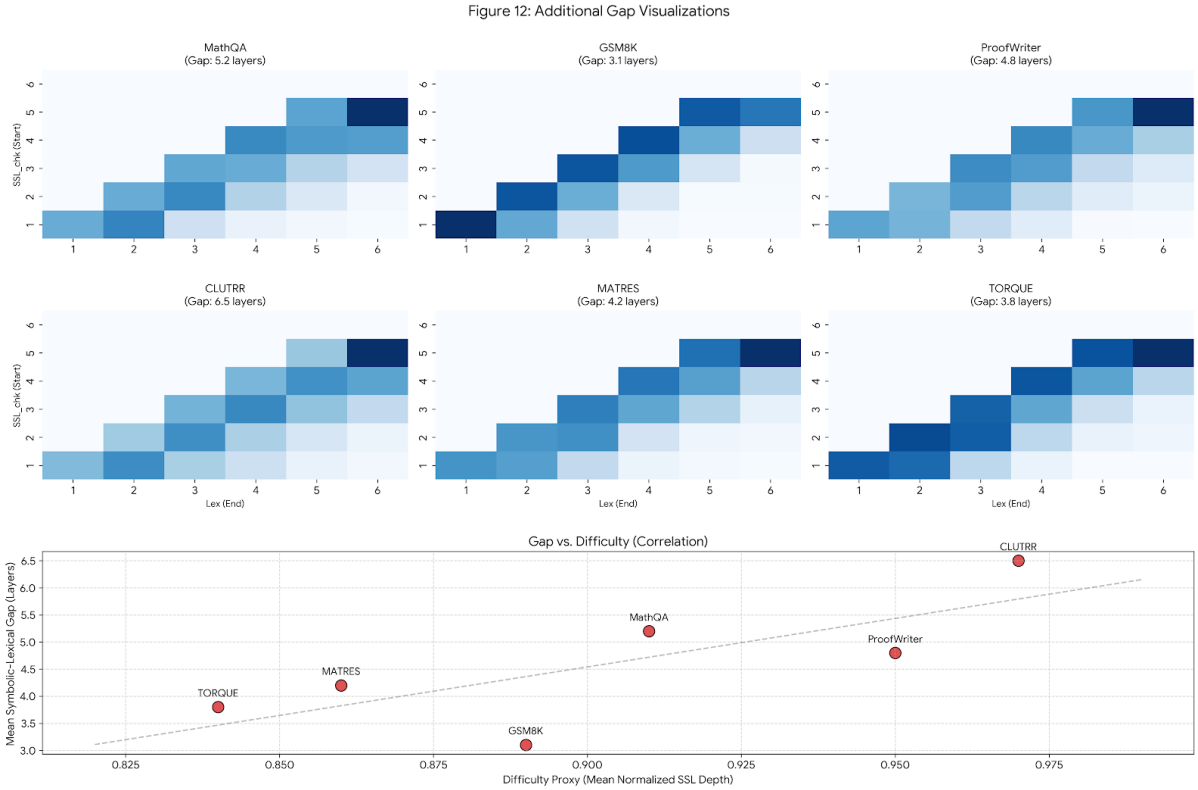

Figure 12: Additional gap visualizations: per-dataset heatmaps and gap–difficulty scatter plots.

Dataset	Trap size (test)	Mean $c(x)$ at ℓ_1	Mean $\tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)/L$
GSM8K	112	0.94	0.89
ProofWriter	88	0.98	0.95
MathQA	254	0.96	0.91
CLUTRR	96	0.92	0.97
MATRES	42	0.95	0.86
TORQUE	55	0.93	0.84

Table 16: ALP trap-set statistics on test. Trap subsets $\mathcal{T}_{\mathrm{trap}}$ are defined by a dev-calibrated confidence threshold at ℓ_1 (top-q; q=0.10), intersected with $\mathrm{Hit}(x, \ell_1)$=0 and either late $\mathrm{SSL}_{\mathrm{chk}}(x)$ or no-SSL. We report the resulting trap size, mean early confidence $c(x)$, and mean normalized sufficiency depth $\tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)/L$ (with $\tilde{\mathrm{SSL}}_{\mathrm{chk}}(x)$=$L$ when $\mathrm{SSL}_{\mathrm{chk}}(x)$= $\perp$). Note the dangerously high early confidence (> 0.90) across all datasets despite the reasoning process being far from complete (SSL > 0.80), highlighting the failure of statistical calibration on complex tasks.

Figure 13: Adversarial Logic Probing (ALP) - Early Confidence vs. Late Certification

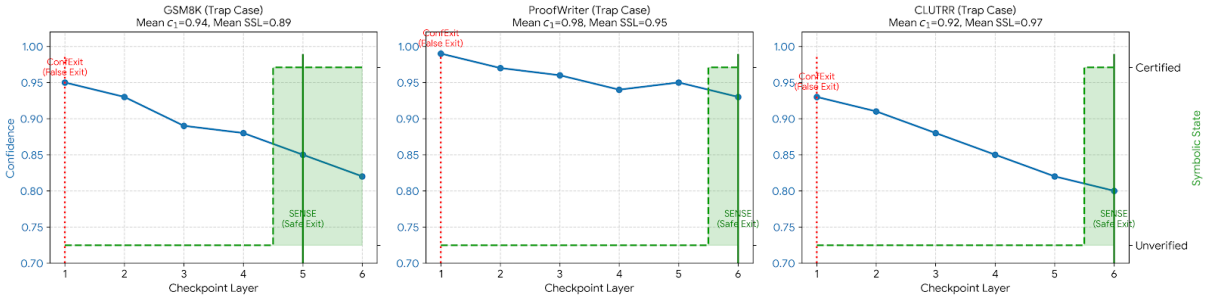

Figure 13: ALP cases: early confidence spikes contrasted with late symbolic certification, and SENSE delaying exit until certification.

Case type	What the instance tests	Expected SENSE behavior	Reported artifacts
Trap	High early confidence, logic incomplete	Delay exit until certification	$c(x)$ curve; Hit; $K(x)$; γ replay
Early-SSL	Clean early closure	Exit at first certified checkpoint	Δs_ℓ; s_ℓ; certificate trace; PRR/CS
No-SSL	Solver timeout / coarse DSL	Fall back to full depth (or safe non-certified exit)	coverage; latency impact; error attribution

Table 17: Case-study reporting format.

Figure 14: Case Studies - SENSE Execution Traces

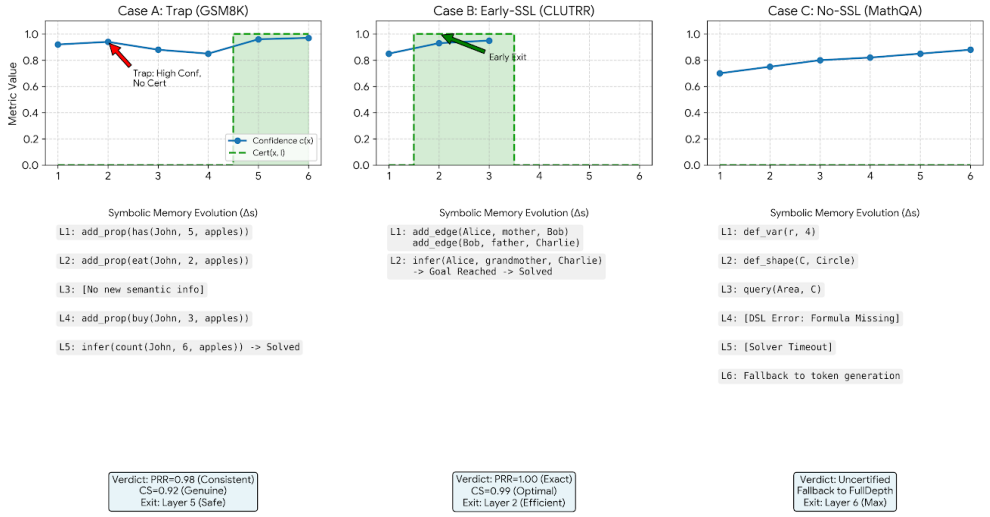

Figure 14: Case-study panel summarizing confidence, certification, and extracted sketches/certificates.

Dataset	Variant	Budget	Perf ↑	EDR ↓	$\text{CVR}_{\text{state}}$ ↓	CertCov@Exit ↑
GSM8K	**SENSE** (Standard)	0.60	**43.2**	0.60	**0.11**	0.75
GSM8K	NoCert (w/o Solver)	0.60	39.5	0.60	0.26	0.58
GSM8K	NoCumul (Memoryless)	0.60	37.8	0.60	0.18	0.52
GSM8K	Coarse $\mathcal{I}$ (Simple)	0.60	38.2	0.60	0.15	**0.88**
GSM8K	Refined $\mathcal{I}'$ (Complex)	0.60	40.5	0.60	0.13	0.62
ProofWriter	**SENSE** (Standard)	0.60	**75.9**	0.60	**0.07**	**0.89**
ProofWriter	NoCert (w/o Solver)	0.60	72.4	0.60	0.21	0.70
ProofWriter	NoCumul (Memoryless)	0.60	66.2	0.60	0.16	0.55

Table 18: Ablations (representative slice at b=0.60). **NoCert** shows that removing the solver degrades safety (CVR spikes), reducing SENSE to a heuristic similar to ConfExit. **NoCumul** highlights the necessity of state persistence for multi-hop reasoning (ProofWriter Perf drops $\sim$10%). **Interface granularity** ($\mathcal{I}$ vs $\mathcal{I}'$) reveals a trade-off: coarse interfaces verify easily (high CertCov@Exit) but miss logic errors, while overly refined ones struggle with coverage.

Figure 15: Qualitative Analyses of SENSE Framework

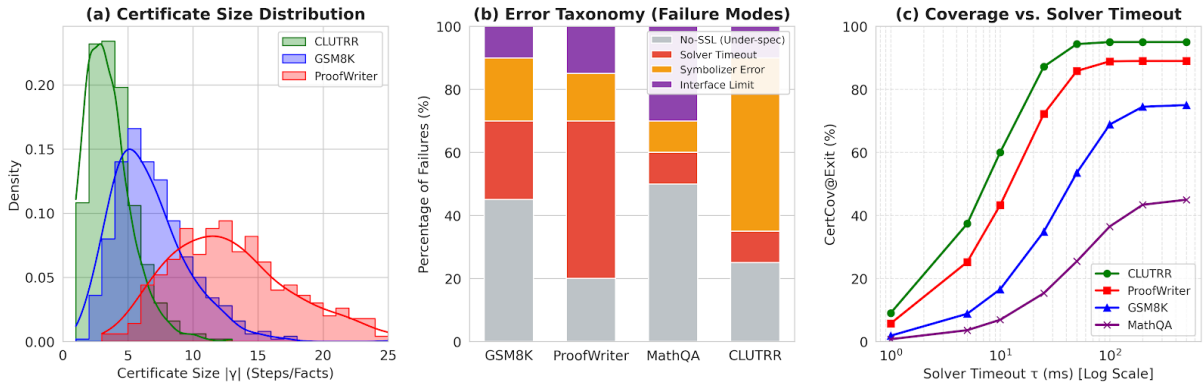

Figure 15: Qualitative analyses: certificate size distributions, error taxonomy, and certification coverage vs. timeout.