What Was Said, Not What Was 'Thought': Type-6 Logic for CoT Verification

Anonymous authors
Paper under double-blind review

Abstract

We introduce Type-6 logic, a variant of dynamic epistemic logic augmented with two operators (uncertainty and recurrence), designed to model the inferential dynamics of contemporary large language model (LLM) chain-of-thought (CoT) reasoning. Type-6 accounts for common LLM reasoning pathologies such as unlicensed revision, enthymemes, loopbacks, and unverifiable/incorrect claims. We propose a verifier based on Type-6 logic that builds a graph out the trace, and checks it against Type-6's axioms and inference rules. We evaluate our framework on LLM-generated CoTs four splits spanning formal and informal reasoning. Our verifier detects structurally unsound reasoning steps that surface-level heuristics miss, and allows for easy visualisation of the model's reasoning process. In our corpus, our verifier shows that derived contradiction is the most common hard-fail category in CoT, and that only about 3% of the propositions of a trace have impact on the final derivation. Ablation studies show that other verification methods (LLMs-as-judges, other neurosymbolic approaches, etc.) cannot be considered interchangeable: for example, agreement between LLMs-as-judges and LINC is $\kappa \approx 0.034$, and this persists within a method across underlying models. Type-6, however, is the most agreed-with method amongst the ones we tested. We prove our verifier runs on average-case linear time; and release our logic specification and artefacts.

1 Introduction

Chain-of-thought (CoT) prompting is now a standard way to induce complex problem-solving capabilities in large language models (LLMs; Wei et al. 2022). This, in theory, means that an LLM's reasoning steps are auditable through their traces. However, CoT lengths mean that they are often intractable for humans to verify. Thus, a body of work has been developed to create automated methods, such as process reward models (PRMs; Lightman et al. 2024; Uesato et al. 2022), prompts (Wang et al., 2023; Madaan et al., 2023), and coherence metrics such as ROSCOE (Golovneva et al., 2023). These methods largely conflate two questions: whether a trace *arrives at* a correct answer, and whether its steps *cohere with one another*. Indeed, it is known that traces often reach the right answer through incoherent steps (Zheng et al., 2025).

We argue that the tokens a model emits do not carry provable truth conditions *for the model itself*: we cannot ascribe knowledge or belief to the generator in any meaningful sense. However, **the statements and conclusions assembled from those tokens do carry truth** conditions *for the reader*. Verification, on this view, lives on the interpretive side of the trace: we do not check what the model knows, but what the trace commits to when read as a natural-language reasoning artefact. Because CoT verification itself is ill-defined (Zaman & Srivastava, 2026), other methods–usually designed for a specific application–often miss this and other pathologies. For example, a trace may commit to p and then, without derivation, commit to $\neg p$–an *unlicensed revision*. It could also conclude a from premises the chain never establishes (an *unverifiable derivation*; distinct from contradiction). Or, it may state a conclusion without ever attempting to derive it (*reasoning avoidance*), or leave key premises doubted or unresolved at the point of concluding. Consider:

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

`K p1` : I know memories are reliable evidence of the past.
`K NOT p1` : Actually, memories could be planted.
`p2` : If memories were planted, we can't trust that yesterday existed.
`THEN a` : So the world was created five minutes ago.

Both the unlicensed knowledge-flip on p_1 (a hard contradiction), and the conclusion a drawn from an antecedent whose grounding depends on an unresolved conditional (an unverifiable derivation), would be missed by answer-correctness and NLI scorers. These are semantic pathologies of reasoning-as-written, and require a more nuanced approach accounting for the fact that CoT traces loop, meander, jump from point to point, and do not necessarily maintain a coherent or sound narrative–regardless of the answer's final correctness.

For this we introduce **Type-6 logic,** a trace-driven update-semantic logic for CoT verification. Type-6 assigns each proposition a state (τ, σ, δ)–truth value, commitment strength (knowledge, belief, or uncommitted), and doubt. It specifies how modal operators, connectives, and structural markers (loopbacks, pivots, doubt) transform that state. We use Type-6 to build a CoT neurosymbolic verifier over a multigraph reconstruction of the trace. This verifier applies Type-6 rules to the reconstruction, and produces three reports: a structural *coherence* report over all reasoning routes from opening to conclusion, a sequential *verification* report logging contradictions and quality signals in order, and a *per-path* report exposing where routes disagree or under-resolve. The design is paraconsistent by construction: contradictions are logged rather than suppressed, and the agent's commitments are always retained. Crucially, Type-6 and its verifier require no gold labels, nor subscribe to a specific concept of CoT verification. Our contributions are:

- **Type-6 logic**: a formal trace-driven update semantics for CoT, with a four-valued truth space, an explicit doubt flag, Kleene-style back-propagation over logical constraints, and a licensed-revision rule that distinguishes epistemic updates from unlicensed reversals.
- **A contradiction taxonomy**: hard and soft contradictions, residual signals (unresolved doubt or unknowability at the conclusion), a trace-level reasoning-avoidance signal, and cross-path disagreement; each mapped to concrete verifier output.
- **A CoT verifier**: a single-pass graph construction equipped with Type-6 logic, producing reports with statement-level provenance.

We test our verifier against four common methods–LLMs-as-judges, ROSCOE, LINC (Olausson et al., 2023), and PRMs–on a human-annotated corpus of 996 formal and informal reasoning traces. We find that none of these methods are interchangeable, since their failure decisions disagree with one another. However, Type-6 is the most-agreed-with method amongst them, correlating with LLM-judges ($\rho = 0.25$) and PRMs ($\rho = 0.31$) more strongly than baseline correlates with the other. We also find with Type-6's verifier deeper insights on traces, such as how many propositions lie on the main derivation chain on average (3%). In all, Type-6 allows for comprehensive, tuneable, and auditable CoT trace verification.

2 Background and Related Work

Scoring chain-of-thought. Outcome-based scorers train on final-answer correctness (Cobbe et al., 2021), or aggregate over samples (Wang et al., 2023). They treat the trace as a means to the answer, scoring two traces identically when one is coherent and the other is not. PRMs instead score individual steps against human annotations (Uesato et al., 2022; Lightman et al., 2024). They are successful at mathematical reasoning (Wang et al., 2024), but require a trained model and do not generalise well beyond their domain. Judgement-based scorers prompt a model to evaluate another's output (Madaan et al., 2023; Shinn et al., 2023; Zheng et al., 2023; Li et al., 2023b). They are flexible, but carry known position and style biases, and are hard to audit at the step level. Reference-free coherence methods like ROSCOE use NLI and embedding metrics to the trace, and have been extended to reasoning-graph metrics (Prasad et al., 2023) and faithfulness probes (Lanham et al., 2023; Turpin et al., 2023). Our work is close to the latter two, but its neurosymbolic nature recovers a parseable structure. It also reports Type-6-based contradictions with statement-level provenance, as opposed to aggregate similarity.

Neurosymbolic and structured-intermediate methods. Other research converts natural-language reasoning into a formal representation and reasons over it symbolically. The most prominent is LINC (Olausson et al., 2023). It translates premises and conclusions into a first-order logic (FOL) and calls a theorem prover. It has had success in arithmetic (Gao et al., 2023; Chen et al., 2023) and constraint satisfaction (Ye et al., 2023) tasks. These approaches build an intermediate structure similar to ours, but typically are restricted to FOL. More importantly, they do not cover–by design–informal reasoning, modal commitments, or approximately-correct derivations. Type-6 preserves these signals, in addition to structural markers and per-path decompositions.

Consistency, hallucination, and self-verification. It is known that CoT as a formal reasoning process is flawed (Zheng et al., 2025; Yueh-Han et al., 2026; Turpin et al., 2023; Lanham et al., 2023); also see Zaman & Srivastava (2026). Approaches hence attempt to check whether the LLM itself is consistent via self-checking prompts (Wang et al., 2023; Manakul et al., 2023), verifier-generator loops (Weng et al., 2023), or entailment probes. These target the generating LLM, while our verifier targets the generated artifact (trace). This distinction determines what the score *means*: when the same trace comes from different generators, or there is no access to the generator, only the artifact-level reading is available. Hallucination detection (Ji et al., 2023; Li et al., 2023a) shares this framing but focuses on factual accuracy rather than inferential structure.

Formal and informal foundations. Reading a reasoning trace as an object of interpretation, distinct from any agent's epistemic state, has roots in informal logic–typically defined as 'anything that is not formal logic' (Perelman & Olbrechts-Tyteca, 1958; Woods & Walton, 1989); also see Blair (2015). Toulmin (2003) argues that everyday reasoning is enthymematic, and Walton (2008) extends this with defeasible argumentation schemes. This is often applied to argument mining (Lawrence & Reed, 2019) and argumentation frameworks (Dung, 1995). Type-6 is more classical, and closer to dynamic epistemic logic (DEL) (Baltag et al., 2016; Van Ditmarsch et al., 2007). It retains modal operators for knowledge and belief, but excludes multi-agent constructs (e.g., announcements) and replaces Kripke-frame semantics with an update semantics indexed by the trace. Still, it remains parametric on a specific system, such as S5 or KD45, since **a CoT does not need to conform to an agent's epistemic profile**, nor it always licenses a coherent Kripke frame. Type-6 also maintains a four-valued discrete truth space $\{T, U_c, U_k, F\}$, to avoid ascribing a subjective credence to the agent. This follows Belnap (1977), , with U_c playing the role of 'no information'; and U_k that of apending free variable. Back-propagation uses strong Kleene semantics (Kleene, 1952) extended so that U_c operands abstain. Paraconsistency (i.e., tolerating local contradiction without global explosion) is by design: Type-6 logs conflicting commitments rather than deriving arbitrary consequences, closer to the annotation-and-log style of Arieli & Avron (1998) than to dialetheic systems (Priest, 2007). This logging means it also admits enthymemes, since its revision rules allow implicit derivations via back-propagation.

3 Type-6 Logic

3.1 Operators and State

Type-6 is a discrete, trace-driven, update-semantic logic meant for CoT verification and audit. It emphasises the fact that CoT tokens do not carry provable truth conditions for the generating LLM, but the statements assembled from them do, for the interpreter. Verification is then at the level of interpretation. This is because Type-6 logic is dynamic: a per-proposition state $\Sigma(\pi) = (\tau, \sigma, \delta)$ assigns a truth value $\tau \in \{T, U_c, U_k, F\}$ (true, unclear/unparseable, unknowable/unresolved, false), a commitment strength $\sigma \in \{\bot, B, K\}$–where K and B are modal operators, discussed below–and a doubt flag $\delta \in \{0, 1\}$ (no/yes). It evolves as the trace is walked, and update rule violations are logged (Section 3.3). This logging is the verifier in disguise. We discuss it in Sections 3 and 4.

Σ evolves via two operator types: **modal** and **structural**, in addition to the standard operators and modifier (NOT) from FOL (Table 1). The modal operators B (belief) and K (knowledge) capture the strength of an agent's commitment to a proposition. This separates statement-level content (what is said, or asserted) from modal commitment (how strongly).

Operator	Agent-side read	Verifier-side update	Notes
K	Claims to know π	$(\alpha, K, 0)$	Requires $\alpha \in \{T,F\}$; clears δ
B	Claims to believe π	$(\alpha, B, -)$	Keeps δ; allows $\alpha = U_k$
?	Doubts its target	$(-, -, 1)$	Only operator setting δ=1
R	Revisits π	No state change	licenses following revision
N	Pivots the discourse	No state change	Inert
NOT	Negates next operator	Flip update's polarity	Folds into K/B/connective
IF	Creates antecedent	Opens an IF-THEN chain	Ends any prior open chain
THEN	Sequential composition	Introduces constraint r	Ends chain; triggers bprop
AND, OR	Conjunction, disjunction	Extends the current chain	Per-operand polarity w/ NOT
(bare)	Assert π, no K,B claim	$(\alpha, -, -)$	Updates τ; leaves σ, δ

Table 1: Type-6 operators and their induced update on $\Sigma(\pi) = (\tau, \sigma, \delta)$. α is the annotation on the current statement; $-$ denotes preservation. Chains are either open (via IF), extended, or closed (via THEN). The latter triggers backprop by introducing a constraint r. N being inert means it preserves the open constraint chain. Modal operators (B, K) commit, but (bare) updates τ without committing. Structural operators $(?, R, N)$ direct the flow.

The structural operators ? (doubt), R (loopback), and N (pivot) govern how the trace revisits, questions, and organises its content without committing to propositional values. They are modelled after the standard CoT discourse. N and R do not change Σ, but change the logging: N preserves an open constraint chain across a pivot, and R marks the next commitment on a proposition as a licensed revision (Section 3.3). ? is the only operator allowed to alter δ: doubt is not an implicit consequence of belief, so $B\,\pi$ (a positive commitment) and $B\,\pi$ followed by $?\,\pi$ (doubted belief) are distinct configurations.

Trace reading. In Type-6, operators are read left-to-right, each scoping over the stack to its right within its statement. To facilitate reading a trace in Type-6, we encode assertions as the *bare* case: a proposition asserted without K or B updates τ but does not change σ. This left-to-right read makes $?\,K\,\pi$ ('do I know π?') and $K\,?\,\pi$ ('I know I doubt π') distinct: the first commits K on π and flags doubt *on the commitment*, and the second flags doubt *on* π then commits K on the doubt-state. At a first glance, doubted knowledge $(K\,?\,\pi)$ and belief $(B\,\pi)$ might look similar, since both yield a value one should not rely on without further support. However, these are semantically distinct, and (σ, δ) preserves this distinction. A worked example is in Appendix D.3.

3.2 Language and Update Rules

Type-6's full description, constraint accumulator, Kleene connective tables, and axiomatic system and metatheory are in Appendix D. From before, remark that every commitment made by an agent can be logged, even when a later derivation suggests it is wrong. Indeed, under Type-6's update and language rules, when a back-propagated constraint assigns π a value conflicting with the agent's committed value, it does not fail. Instead, it yields a derived contradiction but retains the agent's value. In turn, this agent-wins policy allows flagging 'confidently incorrect' outputs–a well-known pathology in CoT.

Language. Let $\mathcal{P}$ be a countable set of proposition symbols. The language $\mathcal{L}_{\text{Type-6}}$ consists of *statements* in two classes:

$$\text{Unary} \quad s = \omega_1 \cdots \omega_k\, \pi : \varsigma, \tag{1}$$

$$\text{Connective-led} \quad s = \beta\, \omega_1 \cdots \omega_k\, \pi : \varsigma, \tag{2}$$

where each $\omega_i \in \Omega_u = \{B, K, ?, R, N, \text{NOT}\}$, $\beta \in \Omega_c = \{\text{THEN}, \text{AND}, \text{OR}, \text{IF}\}$, $\pi \in \mathcal{P} \cup \{\varnothing\}$ is an optional proposition, and ς is an uninterpreted natural-language surface form. A connective-led statement binds π to the proposition of the preceding statement, resolving the connective sequentially. A *trace* is then a finite sequence $R = (s_1, \ldots, s_n)$.

Truth space and state. The truth space $\mathcal{V} = \{T, U_c, U_k, F\}$ is ordered by resolvedness, $U_c \sqsubseteq U_k \sqsubseteq T, F$, with T, F incomparable. U_c (unclear) marks content the interpreter cannot parse, and U_k (unknowable) marks content undetermined but resolvable in principle. The

state is a partial function $\Sigma : \mathcal{P} \to \mathcal{V} \times \mathcal{C} \times \{0, 1\}$ with $\mathcal{C} = \{\bot, B, K\}$. The initial state Σ_0 assigns $(U_k, \bot, 0)$ to every proposition in the trace.

Updates. The modal updates commit a value and a strength,

$$K\,\pi : \Sigma' = \Sigma[\pi \mapsto (\alpha,\ K,\ 0)], \quad \alpha \in \{T, F\}, \tag{3}$$

$$B\,\pi : \Sigma' = \Sigma[\pi \mapsto (\alpha,\ B,\ -)], \tag{4}$$

where α is the annotation on the current statement. A K-commitment with $\alpha{\in}\{U_c, U_k\}$, or a B-commitment with $\alpha{=}U_c$, is a *modal mismatch* (Section 3.3). The K-update clears δ (knowledge is decisive) while B preserves it (belief tolerates residual doubt). A *bare* statement asserting π without a modal operator updates τ and ignores σ. Doubt sets $\delta{=}1$ and preserves the rest; N and R do not alter Σ. Connective-led statements do not act on Σ but extend or close a *constraint chain*: a THEN closes the chain into a constraint $r{\equiv}\phi(p_1, \ldots, p_k)$, which is added to a persistent constraint set and triggers back-propagation.

Back-propagation over the constraint set uses strong Kleene three-valued semantics (Kleene, 1952). We extend them so that a U_c operand makes the enclosing derivation *abstain* rather than propagate (Appendix D.5). Given some $r{\equiv}\phi(p_1, \ldots, p_k)$ and the current state, back-prop either *pins* an unresolved $\tau(r)$ to the derived value, logs a *derived contradiction* when the derived value conflicts with a committed $\tau(r)$ (retaining the agent's value), or logs an *unverifiable derivation* when the operands do not *force* the committed target. Events are logged only on a constraint's status transition, so a single unresolved constraint does not flood the report.

3.3 Updates and Failure Modes

Revisions and reasoning avoidance. An agent may revisit their prior beliefs, as well as skip the evaluation of certain statements. In other words, commitments are defeasible: an earlier value on π may be overwritten, provided the overwrite is *licensed*. A sub-trace between two opposite-valued commitments on π is a *revision segment* iff it contains (i) a jump or loopback edge ,and (ii) a constraint targeting π (explicit or back-propagated). A revision across a revision segment is a *licensed revision* and emits a revision event; without one, it is a K-K contradiction (modal case) or a bare-reassertion conflict (bare case). The two-part condition is core to Type-6's enthymematic characteristics: implicit warrants are permitted, but the verifier requires positive evidence that revision was worked for. Note that licencing captures effort, not correctness: a licensed revision may still be wrong on the merits, which is a matter for downstream evaluation. Revision segments have trace analogues: if no constraint targets the sink proposition a, even transitively, the trace is *reasoning-avoidant*: it announces a conclusion without attempting to derive it. This is a hard trace-level signal, since the absence of any conclusion-directed derivation is a property of the trace rather than of any route through it. Full definitions of the licencing window and its interaction with same-value re-assertion are in Appendix D.

Contradictions and quality signals. Parsing the logs yields two signals: *contradictions*, which flag update-rule violations, and *quality signals*, which flag rhetorical patterns impacting coherence without violating the logic. A summary of these is in Table 2. Hard contradictions are direct violations of the update or back-propagation rules; soft contradictions are weaker inconsistencies that may reflect legitimate epistemic moves under some readings, and are flagged rather than blocking. Residual signals (unresolved doubt or unknowability at the sink) are properties of the terminal state and are elevated to hard when the residual proposition lies on the direct implication chain to the answer.

3.4 From Type-6 to Type-6-based Verification

Type-6 is only a logic. Our CoT verifier works by (1) parsing the trace and applying Type-6 to its statements, and then (2) evaluating these statements through symbolic verification. This induces a natural hierarchical relationship between Type-6 and this verifier. For (1), applying Type-6 means that the labelling is *implicit* in the CoT: a competent interpreter simply surfaces it. In (2), this evaluation necessarily induces a (multi di)graph: propositions become nodes, statements become edges, and structural edges (gap, jump, loopback, meander)

Category	Trigger	Severity
K-K contradiction	$K\,\pi{=}v_1$; $K\,\pi{=}v_2$; no revision segment	†Hard
Derived contradiction	Back-prop yields $v' \neq v$ for committed π	†Hard
Modal mismatch (U_c)	$K\,\pi$ or $B\,\pi$ with $\alpha = U_c$	†Hard
Modal mismatch (U_k/K)	$K\,\pi$ with $\alpha = U_k$	†Hard
Reasoning-avoidance	No constraint targets sink a	†Hard (trace)
B-B conflict	$B\,\pi{=}v_1$; $B\,\pi{=}v_2$; no revision segment	Soft
Bare-reassertion conflict	Bare $\pi{=}v_1$; bare $\pi{=}v_2$; no revision segment	Soft
Unjustified modal shift	$B{\to}K$ opp., $K{\to}B$ same/opp.; no derivation	Soft
Unresolved doubt	$\delta(\pi) = 1$ at sink a	Soft*
Unresolved unknowability	$\tau(\pi) = U_k$ at sink a	Soft*
Cross-path disagreement	Distinct terminal values on π across paths	Soft (global)
Redundant re-assertion	$K\,\pi \to K\,\pi$, no intervening content	Quality
Self-questioned K/B	$K\,\pi \to K\,?\,\pi$ (analog for B)	Quality
Unverifiable derivation	Kleene U_k derivation, committed T/F target	Quality
Ambiguous negated connective	NOT folded into a connective	Quality

Table 2: Type-6 contradiction and quality-signal taxonomy. 'Soft*' severity is elevated to 'hard' when π is in the direct implication chain to a. † marks if the signal blocks the path from being considered coherent for aggregation.

follow from the sequential structure of the trace. Multiple edge-simple paths from source to sink are candidate reasoning routes. The rules by which symbolic verification occurs are Type-6 semantics over the sequential states Σ from the graph. Comparing projected terminal states show two divergence modes: *cross-path disagreement*, where two internally consistent paths yield opposite values on π (the trace is structurally underdetermined), and *under-resolution*, where some paths resolve π and others leave it as U_k. The *direct implication chain* to a is the proposition set reaching a through the transitive closure of constraints along a path. Residual signals on chain propositions are elevated to hard: an unresolved conclusion-supporting premise is not the same as an unresolved aside. This chain is computed per path, so a proposition may be on or off-chain based on the path.

4 Methods and Experimental Setup

4.1 Type-6-based Verification

We now discuss our concrete verifier implementation. For flexibility in application, our main constraint is that a verifier should not be an author (i.e., fill in missing warrants, resolve underdetermined propositions beyond back-prop, or overrule the agent's committed values when they conflict with derived ones). The verifier is then an algorithm which takes in a raw CoT, annotates it in Type-6, and then produces reports from a sequential walk on a multi-digraph built from the annotation. It runs in quasilinear time when accounting for CoT sparsity (Appendix E), and operates in five stages. For a worked example of Type-6, verification and a comparison with baseline outputs, see Appendix B.

Stages 1-3: Type-6 annotation. The verifier (1) prompts an LLM to split and normalise statements, and assign them propositional variables: 'I saw a dog but not a cat' becomes `p1: I saw a dog` and `AND NOT p2: I did not see a cat`. Then (2) function call maps recurring propositions to the same symbol, and (3) an LLM assigns per-statement values from $\{T, U_c, U_k, F\}$ and Type-6 operators to each line-statement using that map. It is also requested to revise any propositions missed. The LLM used was GPT-5 (OpenAI, 2026).

Stages 4-5: verification. The verifier then builds a graph of the CoT applying the edge-kind rules of Section 3.4: each operator becomes an edge, each proposition a node, and consecutive statements yield gap, jump, loopback, or meander edges. Then, it enumerates all paths by computing all edge-simple paths from source (the user prompt) to sink (the answer), subject to a loopback budget. Node revisits are permitted only through jump,

loopback, or binary-connective edges. From the graph and the sequential verifier state, the verifier produces three reports: *coherence*, aggregating structural statistics across paths as distributions (min, max, mean per criteria); *sequential verification*, logging contradictions and quality signals in order and *per-path*, projecting sequential events onto individual paths and tabulating cross-path disagreements and under-resolutions.

4.2 Corpus creation and verifier review

Our corpus is comprised of 996 CoT traces elicited by prompting two Qwen3 models (8B and 17B; Yang et al. 2025) on questions randomly drawn from three sources and two core domains (multiple choice and argumentation). For multiple-choice, we sampled 276 entries from CoT-Logic, a chain-of-thought benchmark;[1] and 370 from MMLU (Hendrycks et al., 2021). For argumentation we collected 311 arguments from DebateLab-KIT's arguments-and-debates collection.[2] In the argumentation subset, the models were instructed to respond to the prompt with a counterargument. In addition, we hand-crafted 39 question-answer pairs covering cases underrepresented in the sampled sources, mixed between argumentation and multiple choice. The median resulting trace length (as line-statements) is 66 for CoT-Logic, 54 for DebateLab, 44 for MMLU, and 52 for the handcrafted subset, for an average of 55.5. Multiple-choice traces probe deductive and eliminative reasoning under formal constraints, while argumentation traces exercise informal reasoning where enthymemes and rhetorical moves dominate. Both span the domains that Type-6 is designed for. Finally, we stripped the CoTs of model-specific tokens (`<think>…</think>` and analogues).

To ensure reliability of Stages 1-3, three annotators trained on Type-6 logic reviewed the annotated traces in full over ten months. They evaluated proposition and operator-and-proposition assignment correctness. If the latter was incorrect, they were asked to provide a correction. Human annotator via Fleiss' κ showed a raw pairwise agreement of κ=0.969 and 0.929 for the first and second criteria. The LLM did not annotate proposition correctness since it is not part of Type-6, but its accuracy on operator-and-proposition assignment was 97.2%, as per the corrections supplied by at least two annotators, split between 98.5% (unverified) and 95.9% (verified). Further details are in Section 7.

4.3 Measurement

We compare against five methods, each covering at least one of the applications of Type-6 and its verifier. More details are in Appendix C. All baselines only work with the natural-language trace, so the comparison is at the level of the raw CoT. This is in line with our verifier's intended use as *post-hoc* evaluator. **ROSCOE** is a reference-free set of metrics. It measures faithfulness, informativeness, repetition, reasoning alignment, chain self-consistency, and step contradiction. We aggregated it to a $[0, 5]$ score with a hard-fail flag on the self-consistency and reasoning-alignment criteria. We use DeBERTa (He et al., 2021) as the NLI model. In **LINC** an LLM translates the trace into FOL, and formal verifier Z3 (de Moura & Bjørner, 2008) checks for premise consistency and entailment. The LLMs used are GPT-5, Gemma-4 (DeepMind, 2026), GLM-4.7 (Z.ai, 2025), and Qwen-3.5. Verdicts are entailment, non-entailment, contradiction (treated as hard-fail), or unknown. We compute translation fidelity on every entry. **PRM**s assigns per-step correctness probabilities from `+`/`−` token logits, and hard-fails when mean or minimum step-probability falls below thresholds. A low-variance mid-range proxy flags likely out-of-distribution operation. The models used are the ones provided by (Wang et al., 2024; Dong et al., 2024), and (Zhang et al., 2025). Finally, we also test **LLMs-as-judges** with a five-criteria rubric (contradiction, unsupported conclusion, modal mismatch, unresolved doubt, and an overall $[0, 5]$ coherence rating). Hard-fails happen when any binary axis fires. The LLMs-as-judges used are GPT-5.6, GLM-4.7, Claude Opus-4.8 (Anthropic, 2026), and Llama-3.1 (Meta, 2024). We additionally consider a **trivial** baseline with four model-free signals: statement count, connective density, adjacent-statement repetition rate, and type-token ratio. Hard-fails on too-short or connective-free traces.

[1] `https://huggingface.co/datasets/isaiahbjork/cot-logic-reasoning`

[2] `https://huggingface.co/datasets/DebateLabKIT/arguments-and-debates`

	LINC	LLM-J	PRM	ROSC.	Trivial	Type-6	LINC	LLM-J	PRM	ROSC.	Trivial	Type-6
LINC	—	0.034	0.004	0.000	-0.005	0.012	0.053	0.050	-0.043	0.024	0.005	
LLM-J	0.004	—	0.067	0.000	0.002	0.151	-0.000	—	0.363	-0.030	0.261	0.248
PRM	0.001	0.054	—	0.000	0.000	0.074	−0.013	0.125	—	-0.404	-0.093	0.312
ROSC.	0.000	0.000	0.000	—	0.000	0.000	-0.054	-0.214	-0.258	—	0.068	-0.255
Trivial	-0.004	0.001	0.000	0.000	—	0.001	−0.024	0.343	-0.169	0.068	—	-0.018
Type-6	0.005	0.092	0.057	0.000	0.001	—	0.035	0.170	0.212	-0.255	-0.018	—

Table 3: Pairwise Cohen's κ for hard-fail agreement (left) and Spearman ρ between trace-level scalars (right), as average over per-method models (blue) and best-of (red). For Spearman ρ, all outputs project to $[0, 5]$; Type-6' scalar is `coherence_score_graded`. LINC's ρ is only for demonstration: our projection took verdict to either error, unknown, contradiction, non-entailment, or entailment, which is not fully comparable with the others.

5 Results

5.1 General Results

We report two criteria as a normalisation factor for all six methods: (1) hard-fail agreement, where each method emits a binary flag per trace (accept/reject as incoherent) and measured with pairwise Cohen's κ; (2) scalar correlation, where each method emits a $[0, 5]$ trace-level coherence score. For the latter our verifier used the coherence score from the report–that is, the direct verifier output–which is the most sensible reduction for this task. Pairwise hard-fail agreement and scalar correlations are in Table 3. Breakdowns by generation model and task type for both metrics are in Appendix F. From the table it can be seen that pairwise hard-fail agreement is vanishingly small ($\kappa \in [-0.005, 0.15]$) among nearly all method pairs. The highest values (LLM-judge-Type-6, $\kappa = 0.15$; LLM-judge-PRM, $\kappa = 0.07$) indicate weak agreement. **There are no methods whose hard-fail decisions can be treated as interchangeable**, but Type-6 is the most agreed-with. This holds when we take best-of or average scores. ROSCOE and Trivial rows show $\kappa \approx 0$, suggesting near-uniform emission of a single label. These verifiers **are labelling different subsets of the traces as failing**, and their disagreement is not due to the underlying model. The scalar correlations support this observation. Some method pairs weakly correlate: LLM-judge-PRM ($\rho = 0.36$ best-of; 0.13 avg.), LLM-judge-Trivial (r. 0.34, < 0), and PRM-Type-6 (r. 0.31, 0.21). ROSCOE anti-correlates with all ($\rho \in [-0.40, -0.03]$), which, combined with its 100% hard-fail rate, indicates it is doing something systematically different from the other methods. Overall, ρ, while non-trivial in a few pairs, **is much lower than expected** were these methods be measuring the same underlying quantity. No pair exceeds $\rho = 0.4$, and most sit below 0.2.

5.2 Ablation: Per-Method Measurements

Our results require a per-baseline breakdown to understand *how* each succeeds or fails. Further details are in Appendix F.2. By construction, Trivial hard-fails almost no traces (0.2%), but correlates non-trivially with LLM-judges ($\rho = 0.34$ with best-model; Table 3), suggesting that **some of what LLM-judges term coherence is captured by trace surface features like length and lexical diversity.** This does not necessarily mean a false-positive, however. Conversely, ROSCOE hard-fails on 100% of the corpus and its scalar anti-correlates weakly-to-moderately with every other method ($\rho \in [-0.40, -0.03]$), despite its internal metrics spanning informative ranges. LINC's performance heavily depends on its translation model. Only GPT-5.6 produces informative verdicts (14.7% entailment, 65.3% non-entailment, 4.0% contradiction hard-fail), but does so from a very partial formalisation (coverage median 0.006). Qwen-3.5-9B and GLM-4.7 emit 85.6% and 72.8% unknown, respectively, and error out otherwise. All PRMs exhibit collapse: Llama-3.1 hard-fails 68.4%; Math-Shepherd 96.2%; Qwen-2.5 100%. They were trained on mathematical reasoning, but our corpus, like most CoT, is mostly natural-language argumentation. Zero low-confidence emissions from Llama and Qwen indicate that these PRMs are confidently mislabelling, rather than expressing uncertainty. LLM-judges grouped in two: capable judges (Claude Opus, GPT-5.6, Gemma-4B) hard-fail 61-68% of traces with moderate co-

herence scores (3.42-4.14) and agree modestly with each other (within-judge κ up to 0.38). The rest hard-fail 98-100% of traces, with pairwise κ below 0.01 against the capable judges. **LLM-as-judge coherence measurements, on this corpus, are dominated by the judge model rather than by the trace.** Through its findings, Type-6 shines as a structural verifier: derived contradiction is the most common hard-fail (409 traces, mean 5.28 events per triggered trace), followed by reasoning-avoidance (548) and unresolved doubt at sink (495). Direct-implication-chain statistics show median on-chain fraction of 0.025 (IQR $[0.016, 0.074]$): **only ∼3% of proposition-level statements in a typical trace directly bear on the derivation of the sink**; the remainder is context, exposition, or restatement. Type-6's alternative graded reductions correlate strongly ($\rho \geq 0.70$; Table 19), indicating this degree of freedom does not qualitatively change trace ordering.

6 Discussion

Our main results and ablation show that CoT verification is brittle beyond definitions or corpus-level agreement, and depends strongly on the model used. Some results can be explained easily: Trivial partially predicts LLM-judge scores through surface features, because these features matter for reasoning. ROSCOE's degenerate hard-fail score captures a real property (its metric distributions are non-degenerate), but unrelated to the other methods' measurements. This is likely due to out-of-distribution effects, which is what we argue also caused PRM's collapse. However, informal logic and argumentation *is* the formalism behind CoT. From this angle, although the failures from PRM and LINC (74.9% of our corpus was marked as 'unknown') are perhaps expected, NLI-based methods should be more successful. True enough, although ROSCOE was insufficient, LLM-judges fared better. Their main failure mode was due to model quality and low–but not zero–performance. Thus, **CoT verification methods are not measuring the same thing** in terms of hard-fail decisions or of scalar scores, as they have large, model-and-data-split invariant disagreements.

This is why we framed Type-6 around structure. Type-6 is not as a better point on the same 'coherence' axis as the baselines, but it operates in a different regime, where the verifier imposes as few assumptions as possible on the trace, while still relying on a formal framework. Still, **the methods are not interchangeable**, but are complementary. For example, it is clear that LLM-judges methods are reasonable defaults for rough, approximate signals, but ultimately statistical and untrustworthy. PRM and LINC are effective within their domains of application. Type-6 overarches–but does not substitute–them in the sense that provides auditability at the statement-level; i.e., it answers '*where* did the reasoning go astray?' or 'is the reasoning good enough?', rather than 'did it fail?'.

7 Conclusion

Type-6 is a discrete, trace-driven update-semantic logic that verifies coherence at the level of the artefact: it treats the agent as the authoritative narrator, records commitments and contradictions with statement-level provenance, and encodes contemporary CoT idiosincrasies like meandering and loopbacks as part of its rules. We built a verifier based on Type-6 and compared it against five baselines (Trivial, ROSCOE, LINC, PRM, LLM-as-judge) on a 996-trace corpus spanning multiple-choice and argumentation. Among these, no pair produces interchangeable failure decisions ($\kappa < 0.4$ across all pairs; $\rho < 0.4$ across all scalar pairs). Type-6 is the most-agreed-with method: it correlates with PRMs ($\rho = 0.31$) and LLM-judges ($\rho = 0.25$) more strongly than those methods correlate with each other, while producing this signal from a stable structural framework rather than from an LLM's judgement of the trace. It also showed the ability to surface structural pathologies that black-box scorers miss entirely, such as reasoning avoidance (present in 548 of 996 traces) and on-chain unresolved commitments–where only ∼3% of a trace's propositions actually drive its conclusion. Although developed for single-agent contemporary CoT, Type-6's update semantics and per-path decomposition extend without modification to multi-agent and structured reasoning systems such as tree-of-thoughts (Yao et al., 2023). Whether Type-6 offers the same performance in that setup than what we observed here; as well as a completeness theorem against a paraconsistent semantic target (e.g., Arieli & Avron 1998), are left for future work.

Ethics Statement

Type-6 verifies the internal coherence of a reasoning trace, not the truth of its conclusions or the intent of its author, and should not be deployed as a factuality checker or a certificate of correctness. Conflating a high coherence score with trustworthiness would misuse the tool in exactly the way this paper argues against. Human annotators were paid $25-47 USD/hour depending on seniority. They worked on a schedule they controlled over the review period. The traces they reviewed were drawn from public benchmarks and hand-crafted items and contain no personal, private, or identifying data. No annotator was asked to review harmful or distressing content; the reasoning traces concern logic problems, academic multiple-choice questions, and philosophical argumentation. Further annotation details are in Appendix C. DebateLab-KIT and MMLU are readily available under ODC-By 1.0 and MIT licenses.

Reproducibility Statement

The verifier is deterministic: given a fixed annotated trace, it produces identical reports on every run, and its behaviour is auditable at statement-level. We provide formal rules, metatheory, and connective semantics in Appendix D. All models, versions, parameters, hardware, and the annotation protocol are in Appendix C. Our logic specification, parser, visualization code, annotation rubric, and annotated dataset are publicly available.[3]

AI Use Statement

Since LLMs are the core object of evaluation, traces and part of the verifier system rely on them. They are also part of the standard SOTA approaches for the field, and so they were used as evaluators in the LINC, PRM, and LLM-as-a-judge baselines. Details on this are documented in Appendix C. An LLM (Claude Opus-4.8) was used to aid in the development of the verifier codebase, and to polish writing. Unit tests were hand designed to ensure correctness of the code, along with manual verification. Writing was reviewed by the authors to ensure conciseness, clarity, and appropriate style. A separate LLM (GPT-5.6 Sol) was used to aid in the development of proofs. The proofs were verified by hand by the authors.

References

Anthropic. Introducing Claude Opus 4.8, 2026. URL `https://www.anthropic.com/news/claude-opus-4-8`.

Ofer Arieli and Arnon Avron. The value of the four values. *Artificial Intelligence*, 102 (1):97–141, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00032-0. URL `https://www.sciencedirect.com/science/article/pii/S0004370298000320`.

Alexandru Baltag, Lawrence S. Moss, and Sławomir Solecki. *The Logic of Public Announcements, Common Knowledge, and Private Suspicions*, pp. 773–812. Springer International Publishing, Cham, 2016. ISBN 978-3-319-20451-2. doi: 10.1007/978-3-319-20451-2_38. URL `https://doi.org/10.1007/978-3-319-20451-2_38`.

Nuel D. Belnap. *A Useful Four-Valued Logic*, pp. 5–37. Springer Netherlands, Dordrecht, 1977. ISBN 978-94-010-1161-7. doi: 10.1007/978-94-010-1161-7_2. URL `https://doi.org/10.1007/978-94-010-1161-7_2`.

J. Anthony Blair. *What Is Informal Logic?*, pp. 27–42. Springer International Publishing, Cham, 2015. ISBN 978-3-319-21103-9. doi: 10.1007/978-3-319-21103-9_2. URL `https://doi.org/10.1007/978-3-319-21103-9_2`.

Ned Block. Troubles with functionalism. *Minnesota Studies in the Philosophy of Science*, 9:261–325, 1978.

[3] Artefacts are at `anonymous.4open.science/r/type6-DC24/`

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

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. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78800-3.

Google DeepMind. Gemma 4, 2026. URL https://gemma4.com/.

Hans Van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF Workflow: From Reward Modeling to Online RLHF. *Transactions on Machine Learning Research*, 2024. URL https://mlanthology.org/tmlr/2024/dong2024tmlr-rlhf/.

Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2): 321–357, 1995. doi: 10.1016/0004-3702(94)00041-X.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=xYlJRpzZtsY.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021. URL https://mlanthology.org/iclr/2021/hendrycks2021iclr-measuring/.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), March 2023. ISSN 0360-0300. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, December 2019. doi: 10.1162/coli_a_00364. URL https://aclanthology.org/J19-4006/.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6449–6464, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.397. URL https://aclanthology.org/2023.emnlp-main.397/.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Meta. Introducing Llama 3.1: Our most capable models to date, 2024. URL https://ai.meta.com/blog/meta-llama-3-1/.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.313. URL https://aclanthology.org/2023.emnlp-main.313/.

OpenAI. Gpt-5.6, 2026. URL https://openai.com/index/gpt-5-6/.

Chaïm Perelman and Lucie Olbrechts-Tyteca. *New Rhetoric, The A Treatise on Argumentation*. University of Notre-Dame Press, 1958.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. ReCEval: Evaluating reasoning chains via correctness and informativeness. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10066–10086, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.622. URL https://aclanthology.org/2023.emnlp-main.622/.

Graham Priest. Paraconsistency and dialetheism. In Dov M. Gabbay and John Woods (eds.), *The Many Valued and Nonmonotonic Turn in Logic*, volume 8 of *Handbook of the History of Logic*, pp. 129–204. North-Holland, 2007. doi: https://doi.org/10.1016/S1874-5857(07)80006-9. URL https://www.sciencedirect.com/science/article/pii/S1874585707800069.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Stephen E. Toulmin. *The Uses of Argument*. Cambridge University Press, 2 edition, 2003.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965. Curran Associates, Inc., 2023. doi: 10.52202/075280-3275. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ed3fea9033a80fea1376299fa7863f4a-Paper-Conference.pdf.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Douglas N. Walton. *Informal Logic: A Pragmatic Approach*. Cambridge University Press, 2008. doi: 10.1017/CBO9780511808630.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations*, 2023. URL https://mlanthology.org/iclr/2023/wang2023iclr-selfconsistency/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2550–2575, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.167. URL https://aclanthology.org/2023.findings-emnlp.167/.

John Woods and Douglas Walton. *What is Informal Logic?*, pp. 221–232. 1989. doi: 10.1515/9783110816082-019.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 45548–45580. Curran Associates, Inc., 2023. doi: 10.52202/075280-1974. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8e9c7d4a48bdac81a58f983a64aaf42b-Paper-Conference.pdf.

Chen Yueh-Han, Robert McCarthy, Bruce W. Lee, He He, Ian Kivlichan, Bowen Baker, Micah Carroll, and Tomek Korbak. Reasoning models struggle to control their chains of thought, 2026. URL https://arxiv.org/abs/2603.05706.

Z.ai. GLM-4.7, 2025. URL https://huggingface.co/zai-org/GLM-4.7.

Kerem Zaman and Shashank Srivastava. Is chain-of-thought really not explainability? chain-of-thought can be faithful without hint verbalization. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 48008–48030, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2217. URL https://aclanthology.org/2026.acl-long.2217/.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Tianshi Zheng, Yixiang Chen, Chengxi Li, Chunyang Li, Qing Zong, Haochen Shi, Baixuan Xu, Yangqiu Song, Ginny Wong, and Simon See. The Curse of CoT: On the Limitations of Chain-of-Thought in In-Context Learning. *Transactions on Machine Learning Research*, 2025. URL https://mlanthology.org/tmlr/2025/zheng2025tmlr-curse/.

A Limitations

Our corpus scope covers multiple-choice and argumentation, with some minor mathematical reasoning questions (word problems). While the patterns found could be arguably extensible due to Type-6's weak assumption set (e.g., no anthropomorphism), the baseline result interpretation could vary. For example, PRM is geared towards arithmetic, not informal, reasoning. On that same vein, the ROSCOE and PRM hard-fail thresholds must be tuned in advance, which means a more complete evaluation would sweep parameters. Do remark, however, that such computational effort defeats the purpose of general-purpose CoT verification.

This verification in Type-6 depends on the LLM annotation pipeline (Stages 1-3). It could be argued that annotation errors propagate into the verifier's reports. However, the high performance of GPT-5 (above 95% accuracy) on their extraction suggests that it is probable this error propagation is not of a concern. What we would argue is a limitation is the dependence on closed models for annotation. For this we have released the corpus, in the hopes that others create and calibrate more extensible solutions. In our work we made an effort to incorporate open and closed models alike for reproducibility purposes.

Finally, our Type-6 characterisation lacks a completeness theorem, even though we established rule-level consistency and paraconsistency by construction. We leave this for future work.

B Two Worked Examples

Table 4 shows a sample Type-6 annotated trace. The graph produced by the verifier is in Figure 1. The trace commits to two propositions (p_{12}, p_{13}). Both carry modal force (Kp_{12}, Bp_{13}). Because the commitments agree with the annotator's labels ($\tau = F$ in both cases, $\alpha \in \{T, F\}$), no modal mismatch and no K-K contradiction fires; Type-6 records the commitments per the *agent as authoritative narrator* policy and moves on. What Type-6 does flag is structural: the trace opens an IF antecedent at line 2 (IF p_9 OR p_{10} AND p_{11}), and interrupted by R without ever being closed by a matching THEN (*malformed implication*). Hence p_9, p_{10} are U_k and remain unresolved at the sink. They lie off the direct implication chain, so they are surfaced as (off-chain) *unresolved-unknowability* residuals rather than elevated to hard. The other methods draw other conclusions: LLM-as-judge flags an unsupported conclusion and modal mismatches on lines accepted by Type-6 as internally consistent; ROSCOE and PRM hard-fail by threshold (r. NLI self-consistency, < 0.5; per-step score < 0.2. LINC errors on translation without producing a verdict. Only Type-6 passes (3.5), penalising the malformed implication but recognising the derivation as structurally sound.

In Figure 2 we show a more theoretical worked example, related to the remaining appendices. It has four canonical patterns from the edge-semantics rules (Appendices **??** and **??**), preserving the same colouring from the applied example from Figure 1. Solid black: logical. Dashed gray: gap. Dotted purple: jump. Dashed orange: loopback. Dashed teal: meander. Blank circles are structural; coloured are propositions (green source, red sink, blue intermediate). In the worst case (Appendix E.2) enumeration is combinatorial in parallel and revisit edges; the cap K and the sampling fallback bound this. In the mostly-linear regime (Appendix **??**), describing realistic CoT graphs, the pipeline is polynomial in trace length and, under typical sparsity, effectively linear. The distributional reporting of coherence, verification, and per-path outputs (Section 4), is a design commitment: coherence is not a scalar, and premature reduction discards the structure the graph exposes.

C Extended Methods

C.1 Models, Hardware, and Parameters

The models used in this work are GPT-5 (reasoning-large), GPT-5.6 (reasoning-sol), GPT-4.1 (gpt-41-longco-2025-04-14) Claude Opus-4.8, Gemma 4 e4B, GLM-4.7-Flash, and Llama

T	`q`	Explain how it relates to other thought experiments in philosophy of mind.
U_k	`IF p9`	If a nation of a billion people can implement (...) a mind,
U_k	`OR p10`	or if the same profile can be implemented in silicon,
T	`AND p11`	and if the phenomenal properties supervene only on this profile,
U_k	`N`	Hmm.
F	`K p12`	Wait, I remember that functionalism entails multiple realizability trivially.
F	`B p13`	Maybe the Chinese Nation is a specific case of that entailment.
U_k	`R`	Let me try to structure this.
U_k	`N`	First, restate the argument.
T	`p14`	The argument shows that a system with (...) is a mind,
T	`AND p15`	and yet nothing about the Chinese Nation intuitively resembles one.
T	`THEN a`	So, functional organization alone is not sufficient for (...) consciousness.

Method	**Hard fail**	**Salient signal**
Trivial	yes	length=12, conn density=0.08
ROSCOE	yes	self-consistency 0.06 $<$ 0.5; mean RA 0.15 $<$ 0.4
LINC	no	errored: no conclusion found.
PRM	yes	min step score 0.13 $<$ 0.2
LLM-judge	yes	unsupported conclusion; modal mismatch; unresolved doubt
Type-6	no	unresolved doubt; malformed implication

Table 4: Method-by-method output on a worked example asking how Block (1978)'s Chinese Nation thought experiment relates to other arguments in philosophy. At the example's leftmost column we note the truth values of each line. These are not used by any of the baselines except Type-6' verifier. LLM-as-a-judge (GPT-5.6) provides a detailed rationale matching Type-6. Remark how each method save LINC (Opus-4.8) and Type-6 hard-fails due to low scoring (e.g. PRM; Math-shepherd). LINC does not hard fail because it fails to find a conclusion and hence it cannot draw a conclusion. Type-6 passes because it tolerates hedged reasoning (p_{13}) and the structural issues are off-chain.

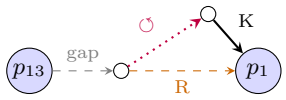


Figure 1: Sample multidigraph generated by the Type-6 verifier on Table 4. The code provided renders source and sink as green and red, respectively, and propositions in blue. Failures are bordered with dashed red and are marked with ✗. Orange borders mark a soft contradiction. Jumps are denoted by ∅.

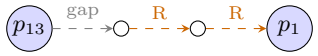
3.1. We used Qwen3 (8B, 17B) for trace generation and Qwen3.5 9B for LINC and LLM-as-a-judge. For ROSCOE we used DeBERTA-large. The GPT-class models as well as Claude were called through the Azure OpenAI API. The rest were called in a consumer-grade computer with 24 CPU cores and 60 GPU cores.

For trace generation and annotation, the Qwen3 models were called with maximum tokens set at 12,000 and temperature 0.8. Outside of this step, all models had their temperature set to zero whenever possible. Proposition extraction, truth-value annotation, and operator assignment were ran primarily on GPT-5 (maximum tokens: 15,000), and GPT-4.1 (maximum tokens=8000) was configured as a fallback for malformed structured outputs. It was invoked on two traces, and so these were discarded due to low annotation quality. The verify-and-correct pass ran also on GPT-5 with the same parameters. Prompts for all passes are in the repository.

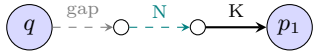
LLMs-as-judges used 1,024 maximum return tokens. LINC models used 4,092 tokens; the version of z3 used is 0.2.0. For ROSCOE, the self-consistency threshold was default at 0.5 and the reasoning alignment threshold at 0.4. Calls were with a batch size of 32. PRM models were set at a 2,048 maximum return tokens. All token parameters were tuned in a 5% subset of the corpus before full usage. High failure rates (e.g., Llama on LINC) meant the model was discarded.



(a) single R with parallel op-chain



(b) chained R: one gap, serial composition



(c) N pivot: gap in, no gap out



(d) gap introduces new material; jump returns

Figure 2: Canonical edge-semantics patterns produced by BuildGraph (Algorithm 2).

D Type-6: Axiomatic Presentation and Metatheory

This appendix presents Type-6 as an axiom system with inference rules, gives a worked example of nested doubt scoping, and proves its basic metatheoretic properties. Throughout, Σ ranges over states, $\mathcal{H}$ over chain states, $\mathcal{K}$ over persistent constraint sets, π over propositions, and v over $\{T, F\}$ unless noted (all as defined in Section 3).

D.1 Judgements

Let $c \in \mathcal{K}$ be $(\pi, \epsilon_t) \equiv \phi(p_1, \ldots, p_k)$ (definitional identity) or $\phi(p_1, \ldots, p_k) \to (\pi, \epsilon_t)$ (implication), where ϕ is a Boolean combination and $\epsilon_t \in \{+, -\}$ the target polarity. Then we say that the constraint set **forces** a value v^*, $\mathcal{K} \models_\Sigma \pi = v^*$, iff evaluating ϕ under Σ via the tables of Appendix D.5 yields a resolved $w \in \{T, U_k, F\}$ (not abstention $*$), with $v^* = w$ if $\epsilon_t = +$ and $v^* = \text{flip}(w)$ if $\epsilon_t = -$.

Here, $\text{flip}(T) = F$, $\text{flip}(F) = T$, $\text{flip}(U_c) = U_c$, and $\text{flip}(U_k) = U_k$. For implications, forcing fires only when ϕ evaluates to T; when $\phi = F$ the implication is vacuously satisfied and forcing does not fire.

Hence, Type-6 has three judgement forms:

- $\Sigma \vdash \pi : (\tau, \sigma, \delta)$ – in state Σ, proposition π carries the indicated triple.
- $(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{s} (\Sigma', \mathcal{H}', \mathcal{K}')$ – statement s transforms the configuration.
- $\mathcal{K} \models_\Sigma \pi = v^*$: the constraint set forces value v^* on π under Σ.

D.2 Inference Rules

We assume a global annotation function α assigning each statement-proposition pair a value in $\mathcal{V}$. Rules operate on $(\Sigma, \mathcal{H}, \mathcal{K})$; untouched components are elided from conclusions. Chain state $\mathcal{H} = (\chi, \phi)$ carries an optional open chain $\chi = ((p_1, \beta_1, \epsilon_1), \ldots)$ of operand triples (proposition, joining connective $\beta_i \in \{\text{seed}, \text{AND}, \text{OR}\}$, and polarity $\epsilon_i \in \{+, -\}$) and a pending-IF flag $\phi \in \{\top, \bot\}$. We write $\chi \cdot (p, \beta, \epsilon)$ for extension and $\varnothing$ for the empty chain.

We now define two crucial operations in Type-6 and its verification.

Definition D.1 (close)**.** Given (χ, ϕ) and a target π with polarity ϵ_t, let $\Psi(\chi)$ be the Boolean combination obtained by evaluating operands left-to-right under AND-binds-tighter-than-OR, with p_i as p_i if $\epsilon_i = +$ else $\neg p_i$, joined by β_i. Then close$(\phi, \chi, \pi, \epsilon_t)$ returns $(\pi, \epsilon_t) \equiv \Psi(\chi)$ if $\phi = \bot$, or $\Psi(\chi) \to (\pi, \epsilon_t)$ if $\phi = \top$.

Definition D.2 (terminate)**.** terminate$(\phi, \chi, \mathcal{K})$ emits a *malformed-implication* event and returns $\mathcal{K}$ if $\phi = \top$ (an unmatched IF is discarded); otherwise returns $\mathcal{K}$ unchanged.

Totality of state. Note that Σ_0 assigns $(U_k, \bot, 0)$ to every proposition in the trace, so every antecedent $\Sigma(\pi) = (\tau_0, \sigma_0, \delta_0)$ below matches a well-defined triple; no rule needs a fresh-proposition case.

Modal rules

$$\frac{\alpha \in \{T, F\}}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{K\,\pi} (\Sigma[\pi \mapsto (\alpha, K, 0)], \mathcal{H}, \mathcal{K})} \text{(K-Commit)}$$

$$\frac{\Sigma(\pi) = (\tau_0, \sigma_0, \delta_0)}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{B\,\pi} (\Sigma[\pi \mapsto (\alpha, B, \delta_0)], \mathcal{H}, \mathcal{K})} \text{(B-Commit)}$$

Figure 3: Modal rules. K-Commit clears δ; B-Commit preserves it. Both leave $\mathcal{H}, \mathcal{K}$ unchanged.

Assertion and structural rules

$$\frac{\Sigma(\pi) = (\tau_0, \sigma_0, \delta_0)}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma[\pi \mapsto (\alpha, \sigma_0, \delta_0)], \mathcal{H}, \mathcal{K})} \text{(Bare-Assert)}$$

$$\frac{\Sigma(\pi) = (\tau_0, \sigma_0, \delta_0)}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{?\,\pi} (\Sigma[\pi \mapsto (\tau_0, \sigma_0, 1)], \mathcal{H}, \mathcal{K})} \text{(Doubt)}$$

$$\frac{}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{R} (\Sigma, (\varnothing, \bot), \text{terminate}(\phi, \chi, \mathcal{K}))} \text{(Loopback)}$$

$$\frac{}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{N} (\Sigma, \mathcal{H}, \mathcal{K})} \text{(Pivot)}$$

Figure 4: Assertion and structural rules. Bare-Assert fires when a statement asserts π with no modal or connective operator in scope; Doubt when ? is outermost. Pivot (N) preserves the configuration. Loopback (R) preserves Σ but resets the chain and terminates any open IF.

Remark that **loopback preserves** Σ. Its role in revision is captured by the presence of an R-edge in the sub-trace, not by any state change; whether a subsequent commitment is a licensed revision is determined by the revision-segment predicate over the trace.

Chain-accumulator rules

$$\frac{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K})}{(\Sigma, (\varnothing, \bot), \mathcal{K}) \xrightarrow{\pi} (\Sigma', ((\pi, \text{seed}, +), \bot), \mathcal{K})} \text{ (Bare-Seed)}$$

$$\frac{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K})}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{\text{AND}\,\pi} (\Sigma', (\chi \cdot (\pi, \text{AND}, +), \phi), \mathcal{K})} \text{ (And-Extend)}$$

$$\frac{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K})}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{\text{OR}\,\pi} (\Sigma', (\chi \cdot (\pi, \text{OR}, +), \phi), \mathcal{K})} \text{ (Or-Extend)}$$

$$\frac{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K}) \quad \mathcal{K}' = \text{terminate}(\phi, \chi, \mathcal{K})}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{\text{IF}\,\pi} (\Sigma', ((\pi, \text{seed}, +), \top), \mathcal{K}')} \text{ (If-Open)}$$

$$\frac{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K}) \quad \Phi = \text{close}(\phi, \chi, \pi, +)}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{\text{THEN}\,\pi} (\Sigma', (\varnothing, \bot), \mathcal{K} \cup \{\Phi\})} \text{ (Then-Close)}$$

Figure 5: Chain-accumulator rules. Each carries a subgoal firing Bare-Assert on the operand or target (Figure 4) and separately updates $\mathcal{H}$. If-Open terminates any prior open IF as malformed; Then-Close constructs Φ via close (Definition D.1) and resets the chain.

Modifier rules

$$\frac{\omega \in \{K, B\} \quad (\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\omega\,\pi} (\Sigma[\pi \mapsto (\alpha, \sigma_1, \delta_1)], \mathcal{H}, \mathcal{K}) \quad \bar{\alpha} = \text{flip}(\alpha)}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\text{NOT}\,\omega\,\pi} (\Sigma[\pi \mapsto (\bar{\alpha}, \sigma_1, \delta_1)], \mathcal{H}, \mathcal{K})} \text{ (NOT-Modal)}$$

$$\frac{\beta \in \{\text{AND}, \text{OR}\} \quad (\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K})}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{\beta\,\text{NOT}\,\pi} (\Sigma', (\chi \cdot (\pi, \beta, -), \phi), \mathcal{K})} \text{ (NOT-Operand)}$$

$$\frac{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\pi} (\Sigma', \mathcal{H}, \mathcal{K}) \quad \Phi^{-} = \text{close}(\phi, \chi, \pi, -)}{(\Sigma, (\chi, \phi), \mathcal{K}) \xrightarrow{\text{THEN NOT}\,\pi} (\Sigma', (\varnothing, \bot), \mathcal{K} \cup \{\Phi^{-}\})} \text{ (NOT-Target)}$$

Figure 6: Modifier rules. NOT-Modal composes with an inner K/B transition and flips the annotation ($\bar{\alpha} = \text{flip}(\alpha)$). NOT-Operand records negated operand polarity; NOT-Target closes with a negated target. NOT-Operand and NOT-Target additionally emit an *ambiguous-negated-connective* quality signal (Section 3).

Back-propagation rules

Back-propagation runs on $\mathcal{K}$ after any transition. Each $c \in \mathcal{K}$ is evaluated against Σ under forcing; at most one rule applies per constraint per step.

$$\frac{\mathcal{K} \models_\Sigma \pi = v^* \quad \Sigma(\pi) = (U_k, \sigma_0, \delta_0) \quad v^* \in \{T, F\}}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\text{backprop}} (\Sigma[\pi \mapsto (v^*, \sigma_0, \delta_0)], \mathcal{H}, \mathcal{K})} \text{ (Backprop-Pin)}$$

$$\frac{\mathcal{K} \models_\Sigma \pi = v^* \quad \Sigma(\pi) = (v, \sigma_0, \delta_0) \quad \{v, v^*\} = \{T, F\}}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\text{backprop}} (\Sigma, \mathcal{H}, \mathcal{K})} \text{ (Backprop-Conflict)}$$

$$\frac{\mathcal{K} \models_\Sigma \pi = U_k \quad \Sigma(\pi) = (v, \sigma_0, \delta_0) \quad v \in \{T, F\}}{(\Sigma, \mathcal{H}, \mathcal{K}) \xrightarrow{\text{backprop}} (\Sigma, \mathcal{H}, \mathcal{K})} \text{ (Backprop-Unverifiable)}$$

Figure 7: Back-propagation rules. Pin promotes an U_k target when its constraint forces a resolved value; Conflict retains the committed value under conflict; Unverifiable applies when the target is committed and the derivation is inconclusive.

Deduplication policy. Note that Backprop-Conflict and Backprop-Unverifiable preserve $(\Sigma, \mathcal{H}, \mathcal{K})$ and additionally emit an event (a *derived-contradiction* and an *unverifiable-derivation*, respectively) governed by the deduplication policy: each $c \in \mathcal{K}$ carries a *last-logged-status* field (initialized $\perp$). Conflict and Unverifiable emit only on *transition* of c's status under Σ: if c's outcome under Σ_i equals its outcome under Σ_{i-1}, no event is emitted at step i.

D.3 Nested Operator Scoping

Operator stacks are read left-to-right, each operator scoping over the stack to its right. For example, $?\,K\,\pi$ ('do I know π?'). Applying $K\,\pi$ (innermost) first, then ?:

$$\Sigma \xrightarrow{K\,\pi} \Sigma[\pi \mapsto (\alpha, K, 0)] \xrightarrow{?} \Sigma[\pi \mapsto (\alpha, K, 1)].$$

The final state is $(\alpha, K, 1)$: knowledge committed on π, but the commitment is under doubt. On the other hand, $K\,?\,\pi$ ('I know I doubt π') applies $?\,\pi$ first, then K–whose target is the doubt-state, not π's value. So K does not update $\sigma(\pi)$; only the inner ? sets the flag:

$$\Sigma \xrightarrow{?\,\pi} \Sigma[\pi \mapsto (\tau_0, \sigma_0, 1)] \xrightarrow{K} \Sigma[\pi \mapsto (\tau_0, \sigma_0, 1)].$$

Hence the final state is $(\tau_0, \sigma_0, 1)$: the knowledge commitment attaches to the doubt, registered via $\delta = 1$; and the value on π is untouched.

D.4 Metatheoretic Properties

Proposition D.3 (Determinism)**.** *For any configuration and statement s, at most one rule of Figures 3 and 6 applies, and the resulting configuration is unique. Back-propagation (Figure 7) then fires deterministically: at most one of Pin, Conflict, Unverifiable applies per constraint per step.*

Proof. The rules partition by the outermost operator of s: K-Commit iff $s = K\,\pi$; B-Commit iff $s = B\,\pi$; Bare-Assert iff s is fully bare; Doubt iff $s = ?\,\pi$ with no modal to its right; Pivot iff $s = N$; Loopback iff $s = R$; the NOT-rules iff the outermost operator is NOT in the corresponding position; the chain rules on connective-led statements. These are mutually exclusive by the grammar of $\mathcal{L}_{\text{Type-6}}$. Each yields an explicit functional update of $\Sigma, \mathcal{H}, \mathcal{K}$; NOT-rules recurse on the inner operator, unique by induction on operator depth. For back-propagation, the three rules partition on disjoint committed-value conditions: $\tau(\pi) = U_k$ (Pin), $\tau(\pi) \in \{T, F\}$ with $v^* \in \{T, F\}, v^* \neq \tau(\pi)$ (Conflict), $\tau(\pi) \in \{T, F\}$ with $v^* = U_k$ (Unverifiable). Hence at most one applies per constraint per step. □

Proposition D.4 (Monotonicity of resolution)**.** *For any trace and π, the resolvedness of $\tau(\pi)$ is nondecreasing in $\sqsubseteq$ across* $\Sigma_0, \Sigma_1, \ldots$ modulo licensed revision*: absent a K-revision or bare-revision at step $i+1$, $\tau_i(\pi) \sqsubseteq \tau_{i+1}(\pi)$.*

Proof. By cases. Modal/bare commits on $\pi' \neq \pi$, Doubt, Pivot, Loopback, and N preserve $\tau(\pi)$. Chain rules carry a Bare-Assert subgoal, so reduce to that case. NOT-Modal composes with an inner modal rule; polarity flip does not change τ's position in $\sqsubseteq$ (both T, F maximal). Backprop-Pin promotes $U_k \sqsubseteq v^*$ (a strict promotion); Backprop-Conflict and Backprop-Unverifiable retain the committed value. □

Proposition D.5 (Local consistency)**.** *No single rule application of Figures 3 and 7 introduces a contradiction in the sense of Section 3, unless the contradiction is explicitly detected and logged by the verifier.*

Proof. By case analysis:

K-Commit: the side condition $\alpha \in \{T, F\}$ excludes U_c, U_k; those cases fire the logged modal-mismatch predicate rather than proceeding silently. A prior differing K-value with no revision segment fires the K-K contradiction; otherwise it is a licensed K-revision and nothing is logged.

B-Commit: permitted on U_c (logged as modal mismatch) and U_k (permitted). A prior differing B-value with no revision segment fires a logged B-B conflict; unjustified modal shifts ($B \to K$, $K \to B$ without an intervening derivation) are logged as soft.

Bare-Assert: permitted on all four values. A prior differing bare value with no revision segment fires the logged bare-reassertion conflict; otherwise a licensed bare-revision, nothing logged.

Doubt, Pivot: do not modify $\Sigma, \mathcal{H}, \mathcal{K}$ and cannot introduce K-K, B-B, or modal-mismatch contradictions.

Loopback: does not modify Σ; resets $\mathcal{H}$ and may fire the malformed-implication event via terminate, logged as a diagnostic, not a contradiction.

Chain rules: f ire their Bare-Assert subgoal (consistency reduces to that case); If-Open with a nonempty prior IF fires the malformed-implication event, logged, without silently corrupting $\mathcal{K}$.

NOT-Modal: reduces to the inner rule (consistency by induction). NOT-Operand and NOT-Target additionally fire the ambiguous-negated-connective quality signal (non-blocking).

Backprop: Pin promotes U_k under a forcing constraint, no contradiction; any later inconsistency with a subsequent commit is detected at that later step. Conflict logs a derived-contradiction and retains the committed value; Unverifiable logs its event without modifying Σ. All three track status transitions per the deduplication policy.

In every case the rule leaves Σ consistent, or the inconsistency is detected and logged before the update completes. This concludes the proof. □

Remark D.6 (Completeness)**.** Type-6 is deliberately *not* complete with respect to a classical propositional or modal semantics: the four-valued truth space and the tolerance of enthymemes preclude a completeness theorem of the DEL variety. Establishing completeness would require fixing a target semantics (e.g., a paraconsistent bilattice in the spirit of Belnap 1977). We leave that for future work.

Remark D.7 (Paraconsistency)**.** A K-K contradiction on π does not license arbitrary further conclusions: contradictions are logged and localized to π, and inference on other propositions is unaffected. **Type-6 is thus paraconsistent by construction**. This is appropriate for CoT traces, which are frequently locally inconsistent without being globally uninformative.

D.5 Semantics of Logical Connectives

In this section we discuss the truth tables used by back-propagation over THEN-led constraints. Operands draw from $\mathcal{V} = \{T, U_c, U_k, F\}$; U_c propagates as a computational *abstention* ($*$): any U_c operand makes the enclosing derivation abstain rather than propagate.

Negation on the underdetermined values is the identity, $\neg U_c = U_c$, $\neg U_k = U_k$, consistent with the no-op polarity flip in NOT-Operand and NOT-Target.

$\wedge$	T	U_k	U_c	F
T	T	U_k	$*$	F
U_k	U_k	U_k	$*$	F
U_c	$*$	$*$	$*$	F
F	F	F	F	F

$\vee$	T	U_k	U_c	F
T	T	T	T	T
U_k	T	U_k	$*$	U_k
U_c	T	$*$	$*$	$*$
F	T	U_k	$*$	F

$\rightarrow$	T	U_k	U_c	F
T	T	U_k	$*$	F
U_k	T	U_k	$*$	U_k
U_c	T	$*$	$*$	$*$
F	T	T	T	T

Entries $*$ mark configurations where a U_c operand renders the constraint uninterpretable at the value level; back-propagation abstains and logs no value-level event. Note F dominates U_c in $\wedge$, T dominates U_c in $\vee$, and F dominates U_c in $\rightarrow$ (vacuous implication). The strong-Kleene short-circuit conditions extended with U_c as an abstaining fourth value. The $\rightarrow$ table is derivable from $\vee$ via $\neg p \vee q$ under negation-as-identity on U_c/U_k. These tables suffice for THEN-led constraints; k-ary generalizations compose associatively.

E Complexity Analysis

This appendix bounds the parsing, construction, verification, and path-analysis phases of the verifier. We prove general worst-case bounds (Theorem E.1) and tighter bounds under the structural regularities of realistic CoT traces (the 'mostly linear' regime;Theorem E.14). It also introduces the pseudocode version of our verifier.

E.1 Preliminaries and Notation

Following Section 3, a CoT trace is a finite sequence $R = (s_1, \ldots, s_n)$ with each

$$s_i = \omega_{i,1}\, \omega_{i,2} \cdots \omega_{i,k_i}\ \pi_i\ :\ \varsigma_i,$$

where $\omega_{i,j} \in \Omega = \Omega_u \cup \Omega_b \cup \{\mathrm{NOT}\}$, π_i is an optional proposition in $\mathcal{P}$, and ς_i is an uninterpreted sentence. Operators partition into unary $\Omega_u = \{B, K, N, ?, R\}$ and binary $\Omega_b = \{\mathrm{THEN}, \mathrm{AND}, \mathrm{OR}, \mathrm{IF}\}$, with NOT a modifier.

Our verifier takes in $n = |R|$ number of statements; t maximum tokens in any single statement with $T = \sum_i (k_i + [\pi_i \neq \varnothing])$ total token count; and $p = |\mathcal{P}_R|$ distinct propositions.

Construction of the verifier yields a labelled multi-digraph $G = (V, E, \lambda)$ with $V = V_p \sqcup V_b$ (proposition nodes bijective with propositions, blank nodes separating consecutive operators). Every operator is an edge; parallel edges are preserved. Each e carries a label $\lambda(e) \in \Omega$ and a kind $\kappa(e) \in \{\mathsf{logical}, \mathsf{gap}, \mathsf{jump}, \mathsf{loopback}, \mathsf{meander}, \mathsf{unknown}\}$.

This graph is parametrised by $|V|, |E|$ (node and edge counts). We denote $\Delta = \max_v \deg^+(v)$ the maximum out-degree and $\mu = \max_{u,v} |\{k : (u, v, k) \in E\}|$ its maximum edge multiplicity. We also assign L loopback budget (`max_loopbacks`, default 1), J number of revisit edges (jump, loopback, and binary-connective edges combined). Crucially, we also cap the enumeration at K (`max_paths`), and the sample count at S; ℓ: sampling walk-length cap.

Edge operations. To build and traverse the graph, there are six edge insertion rules. Let s_i be the current statement, and u the node reached after s_{i-1}. The operations are then:

- **Logical.** An explicit known operator produces a logical edge (adjacent NOT folded in).
- **Gap.** Two consecutive statements with no binary connector, whose target introduces *new* material, are joined by a gap. Gaps are neutral.
- **Jump.** Same condition as gap, but the target *revisits* a seen proposition.
- **Loopback.** Emitted by R; target is the proposition of the next statement carrying one. Chained Rs compose serially.
- **Meander.** Emitted by N; marks a pivot, enforcing a gap into N but not out of it when the following statement carries operators.
- **Unknown.** Any unrecognized token becomes a labelled unknown edge, with a warning.

Algorithm 1 ParseLine(ℓ)

1: split ℓ on `:' into (left, ς)
2: tokenize left into $\tau_1, \ldots, \tau_k$
3: **if** τ_k ends in `?' and $\tau_k \notin \Omega$ **then**
4: strip trailing `?'; append ? to the op list
5: **end if**
6: **if** $\tau_k \in \Omega$ **then**
7: $\pi \leftarrow \varnothing$; ops $\leftarrow \tau_{1..k}$
8: **else**
9: $\pi \leftarrow \tau_k$; ops $\leftarrow \tau_{1..k-1}$
10: **end if**
11: fold each NOT into its preceding operator
12: **return** (ops, π, ς)

The gap/jump distinction is central to the mostly-linear analysis: both arise without a connective, but gap introduces (forward, fresh material) while jump returns (backward, prior material).

E.2 General Worst-Case Bounds

In this section we prove the following:

Theorem E.1 (Worst-case total)**.** *The full pipeline runs in*

$$T_{\text{total}} = O(nt) + T_{\text{verify}} + T_{\text{paths}} + O(N \cdot \bar{p})$$

steps, with $T_{\text{verify}} = O(n^2 t)$, T_{paths} either T_{enum} or T_{sample}, N the number of scored paths, and $\bar{p}$ the mean path length; space $S_{\text{total}} = O(nt + N \cdot \bar{p})$.

For this we need to prove parsing and construction costs (Appendix E.2.1), and the cost of path enumeration and sampling (Appendix E.2.2); as well as back-propagation (Appendix E.2.3).

E.2.1 Parsing and Construction

We prove the following about graph construction (Algorithm 2, BuildGraph) and its parsing subroutine (Algorithm 1; ParseLine):

Lemma E.2 (Graph construction)**.** *BuildGraph (Algorithm 2) runs in $O(nt)$ time (excluding verifier work, bounded in Appendix E.2.3) with $|V|, |E| \in O(nt)$.*

Moreover, when $t = O(1)$–the typical CoT regime–construction is $O(n)$ time and space, with $|V|, |E| \in O(n)$.

Proof. For this, remark that ParseLine runs in $O(t)$ and produces at most t operators. Over n statements, this provides the necessary bound.

A minor nuance is that the adds (e.g., lines 6, 11, 14, etc.) are also constant time and add ≤ 1 blank and ≤ 1 edge. Given the operator list bound of $O(t)$, each statement contributes $O(t)$ nodes and edges and $O(t)$ work (with $O(1)$ bookkeeping for seen, meander, Rtail, source/sink). Summing gives $O(nt)$ and $|V|, |E| \in O(nt)$; post-loop source/sink resolution is $O(1)$. □

E.2.2 Path Enumeration and Sampling

In this section we prove the following:

Lemma E.3 (Enumeration cost)**.** *Let $N_{\text{emit}} = \min(K, N_{\text{total}})$. Then EnumeratePaths (Algorithm 3) runs in*

$$T_{\text{enum}} = O(N_{\text{emit}} \cdot (|V| + L + J) \cdot \Delta + W_{\text{dead}}),$$

with W_{dead} the frame work on non-yielding branches, in $O(|V| + L + J)$ auxiliary space.

Algorithm 2 BUILDGRAPH(R)

initialize empty G; prev $\leftarrow \varnothing$; seen $\leftarrow \emptyset$
meander $\leftarrow \bot$; Rtail $\leftarrow \varnothing$; initialize verifier state vs
for $i \in 1..n$ **do**
  $(\text{ops}, \pi, \varsigma) \leftarrow$ PARSELINE(s_i)
  if $\pi \notin$ seen and $\pi \neq \varnothing$ **then**
    add node π; seen $\leftarrow$ seen $\cup \{\pi\}$; revisit $\leftarrow \bot$
  else if $\pi \neq \varnothing$ **then**
    revisit $\leftarrow \top$
  end if
  if Rtail $\neq \varnothing$ and $\pi \neq \varnothing$ **then**
    add loopback Rtail $\rightarrow \pi$; Rtail $\leftarrow \varnothing$; revisit $\leftarrow \top$
  end if
  if s_i is a single-R statement and Rtail $\neq \varnothing$ **then**
    add blank b'; add loopback Rtail $\rightarrow b'$; Rtail $\leftarrow b'$; **continue**
  end if
  if ops $= \varnothing$ **then** ▷ bare proposition
    if revisit **then** add jump prev $\rightarrow \pi$
    else add gap prev $\rightarrow \pi$
    end if
  else ▷ statement carries operators
    needsGap $\leftarrow$ (prev $\neq \varnothing$) $\wedge$ ($\text{ops}_1 \notin \Omega_b$) $\wedge \neg$revisit $\wedge \neg$meander
    $c \leftarrow$ prev
    if needsGap **then** add blank b; add gap prev $\rightarrow b$; $c \leftarrow b$
    else if revisit **then** add blank b; add jump prev $\rightarrow b$; $c \leftarrow b$
    end if
    for $j \in 1..|\text{ops}|$ **do**
      if $j = |\text{ops}|$ and $\pi = \varnothing$ and $\text{ops}_j = R$ **then**
        Rtail $\leftarrow c$; **break**
      end if
      $t \leftarrow \pi$ if last op and $\pi \neq \varnothing$, else new blank
      add edge $c \xrightarrow{\text{ops}_j} t$ of appropriate kind; $c \leftarrow t$
    end for
  end if
  VERIFIERSTEP(vs, ops, π, α_i) ▷ Appendix E.2.3
  prev $\leftarrow \pi$ if $\pi \neq \varnothing$ else c
  meander $\leftarrow$ (ops $\neq \varnothing \wedge \text{ops}_{\text{last}} = N$)
end for
resolve source and sink; fail if unresolved
return (G, vs)

In Type-6's verifier, coherence is measured over *edge-simple* source-to-sink paths (Algorithm 3). Parallel edges yield distinct paths, and three policies constrain enumeration. These are:

1. **Sink absorption**: once the sink is reached, its out-edges are ignored
2. **Loopback budget**: at most L loopback edges per path
3. **Restricted revisits**: a node may be re-entered only via a jump, loopback, or Ω_b-labelled edge, never via a forward logical or gap edge.

Policy (3) preserves legitimate revisits (jumps, loopbacks, connective returns) as distinct routes while pruning spurious forward-edge revisits, resolving both false-positive enumeration (double-counting sequential progress) and false-negative pruning.

Hence we need to account for these ahead of proving Lemma E.3:

Lemma E.4 (DFS frame cost and path length)**.** *A single DFS frame at v in Algorithm 3, excluding recursion, performs $O(\deg^+(v))$ work. Moreover, every emitted path has length*

at most $|V| - 1 + L + J$, so recursion depth is $O(|V| + L + J)$, where J counts jump *and Ω_b-labe;led edges.*

Proof. A frame at v iterates over $\deg^+(v)$ out-edges, each with $O(1)$ membership tests, kind/label lookups, a budget comparison, and set insert/removal. Moreover, DFS refuses to re-enter a node in V_v unless it is the sink or the edge is jump, loopback, or Ω_b-labeled. Non-sink fresh-node entries are $\leq |V|-1$; loopback traversals are capped at L, and jump/connective traversals are edge-simple, so $\leq J$. Hence $|\pi| \leq |V| - 1 + L + J$. □

Lemma E.5 (Simple-path count)**.** *The number of edge-simple source-to-sink paths in a multi-digraph on $|V|$ nodes with maximum multiplicity μ, respecting the three policies, is*

$$N_{\text{total}} \leq \mu^{|V|-1+L+J} \cdot \sum_{k=0}^{|V|-2+L+J} \frac{(|V| + L + J - 2)!}{k!} = O\Big(\mu^{|V|+L+J} \cdot (|V| + L + J)! \cdot e\Big)\,.$$

Proof. By Lemma E.4, any path has length $\leq |V| - 1 + L + J$. In the underlying simple digraph (parallel edges collapsed), a simple path of edge-length $k+1$ is an ordered choice of k distinct intermediates from $|V|+L+J-2$ positions, $\leq (|V|+L+J-2)!/k!$ choices; summing over k gives the numerator, bounded by $(|V|+L+J-2)! \cdot e$. Restoring multiplicities, each of $\leq |V|-1+L+J$ transitions selects one of μ parallel edges, giving $\mu^{|V|-1+L+J}$. Multiplying gives the claim. □

Loopback budget. Remark that L does not reduce the leading factorial: an adversarial graph realizes the bound with paths containing no loopback edges, so the budget is never engaged. It does cap the length contributed by loopback revisits (Lemma E.4) and is essential to termination when loopback edges are present.

Lemma E.6 (Sampling cost)**.** *SamplePaths (Algorithm 4) runs in $O(S \cdot \ell_{\max}^2 \cdot \Delta)$ time and $O(\ell_{\max})$ auxiliary space per walk.*

Proof. Each walk runs $\leq \ell_{\max}$ steps; each step iterates $\leq \Delta$ out-edges, testing membership against a partial path of length $\leq \ell_{\max}$ ($O(\ell_{\max})$), with $O(1)$ policy and budget checks and $O(1)$ selection. Per-step cost $O(\Delta \cdot \ell_{\max})$, per-walk $O(\ell_{\max}^2 \cdot \Delta)$, S walks $O(S \cdot \ell_{\max}^2 \cdot \Delta)$; the $50S$ attempt cap is a constant factor. Sampled paths are drawn proportional to the product of uniform local choices. This is a well-defined (but in general biased) estimator relative to uniform over N_{total}. □

We are now ready to prove Lemma E.3.

Lemma E.7 (Enumeration cost)**.** *Let $N_{\text{emit}} = \min(K, N_{\text{total}})$. Then EnumeratePaths (Algorithm 3) runs in*

$$T_{\text{enum}} = O(N_{\text{emit}} \cdot (|V| + L + J) \cdot \Delta + W_{\text{dead}})\,,$$

with W_{dead} the frame work on non-yielding branches, in $O(|V| + L + J)$ auxiliary space.

Moreover, ignoring K, N_{total} satisfies Lemma E.5 and

$$T_{\text{enum}} = O\Big(\mu^{|V|+L+J} \cdot (|V| + L + J)! \cdot (|V| + L + J) \cdot \Delta\Big)\,.$$

Proof. From Lemma E.4 we know that each yielded path has length $\leq |V| - 1 + L + J$, produced by a branch of that depth with $O(\Delta)$ per frame. Charging frame work to terminal yields gives the first term, with non-yielding work W_{dead}. Auxiliary state is the recursion stack plus E_v, V_v, all of which are $O(|V| + L + J)$. The worst-case bound follows. □

Algorithm 3 EnumeratePaths$(G, \text{src}, \text{snk}, K, L)$

$Y \leftarrow 0$
procedure DFS(u, E_v, V_v, ℓ)
  if $Y \geq K$ **then return**
  end if
  if $u = \text{snk}$ and $E_v \neq \emptyset$ **then**
    yield E_v; $Y \leftarrow Y + 1$; **return**
  end if
  for $(u, v, k) \in$ out-edges(u) **do**
    if $(u, v, k) \in E_v$ **then continue**
    end if
    if $v \in V_v$ and $v \neq \text{snk}$ and $\kappa((u, v, k)) \notin \{\mathsf{jump}, \mathsf{loopback}\}$ and $\lambda((u, v, k)) \notin \Omega_b$ **then**
      continue
    end if
    $\ell' \leftarrow \ell + [\kappa((u, v, k)) = \mathsf{loopback}]$
    if $\ell' > L$ **then continue**
    end if
    DFS$(v, E_v \cup \{(u, v, k)\}, V_v \cup \{v\}, \ell')$
  end for
end procedure
DFS$(\text{src}, \emptyset, \{\text{src}\}, 0)$

Algorithm 4 SamplePaths$(G, \text{src}, \text{snk}, S, L, \ell_{\max})$

for $s \in 1..S$ **do**
  $u \leftarrow \text{src}$; path $\leftarrow [\,]$; $V_v \leftarrow \{\text{src}\}$; $\ell \leftarrow 0$
  for $t \in 1..\ell_{\max}$ **do**
    if $u = \text{snk}$ and path $\neq [\,]$ **then**
      yield path; **break**
    end if
    $F \leftarrow$ feasible out-edges of u under the three policies
    if $F = \emptyset$ **then break**
    end if
    draw $(u, v, k) \sim \text{Unif}(F)$; append to path; update V_v, ℓ, u
  end for
end for

E.2.3 Verifier Cost

The verifier alternates within construction: after each statement, VerifierStep updates vs, extends any constraint chain, and runs back-propagation.

In this section we prove the following:

Lemma E.8 (Amortised verifier cost)**.** *Across the trace, VerifierStep costs*

$$T_{\text{verify}} = O(nt + n \cdot |\mathcal{K}| \cdot \kappa_{\max}) = O(n^2 t),$$

where $|\mathcal{K}| = O(n)$*,* $\kappa_{\max} \leq t$*; when* $t = O(1)$*,* $O(n^2)$*.*

For this we rely on the following propositions:

Proposition E.9 (Update costs)**.** *For a statement of* $k \leq t$ *operators, the sequential update (K-/B-Commit, Bare-Assert, Doubt, Pivot, Loopback, or a NOT variant) runs in* $O(t)$*; quality-signal checks are* $O(1)$ *using the tracked history.*

Moreover, chain updates (Bare-Seed, And/Or-Extend, If-Open, Then-Close, N-preservation, R-reset) run in $O(k) \leq O(t)$*.*

Proof. Each modal/bare update touches one entry of Σ via $O(1)$ lookup and appends a constant-size event; the NOT fold is resolved at parse time (Lemma E.2). Doubt/Pivot/Loopback flip a flag or leave Σ unchanged in $O(1)$. Iterating the k operators is $O(k) \leq O(t)$. Quality checks compare against the prior commitment on the same proposition, $O(1)$ from vs.

For the chain updates, note that seed and extensions append a constant-size triple; If-Open resets after $O(1)$ terminate; Then-Close builds a constraint via close in $O(|\chi|)$, with $|\chi| \leq k$. Iterating the k operators is $O(k)$. □

Proposition E.10 (Back-propagation per statement)**.** *With $|\mathcal{K}_i|$ the constraint-set size after s_i and $\kappa_{\max}$ the maximum chain length, back-propagation at step i runs in $O(|\mathcal{K}_i| \cdot \kappa_{\max})$.*

Proof. Each $c \in \mathcal{K}_i$ is evaluated by a scan of its operand chain ($\leq \kappa_{\max}$, each lookup $O(1)$); the three-way case split and $O(1)$ event emission with $O(1)$ deduplication check follow. Summing over c gives the claim. □

We are now ready to prove Lemma E.8:

Lemma E.11 (Amortised verifier cost)**.** *Across the trace,* VerifierStep *costs*

$$T_{\text{verify}} = O(nt + n \cdot |\mathcal{K}| \cdot \kappa_{\max}) = O(n^2 t),$$

where $|\mathcal{K}| = O(n)$, $\kappa_{\max} \leq t$; when $t = O(1)$, $O(n^2)$.

Proof. Immediate from Proposition E.9 and Proposition E.10. The former gives $O(t)$ per statement, $O(nt)$ total. The latter gives $O(|\mathcal{K}_i| \cdot \kappa_{\max})$ per step; summing with $|\mathcal{K}_i| \leq |\mathcal{K}| = O(n)$ gives $O(n^2 t)$. □

E.3 Proof of Theorem E.1

The proof follows:

Theorem E.12 (Worst-case total)**.** *The full pipeline runs in*

$$T_{\text{total}} = O(nt) + T_{\text{verify}} + T_{\text{paths}} + O(N \cdot \bar{p})$$

steps, with $T_{\text{verify}} = O(n^2 t)$, T_{paths} either T_{enum} or T_{sample}, N the number of scored paths, and $\bar{p}$ the mean path length; space $S_{\text{total}} = O(nt + N \cdot \bar{p})$.

Proof. Parse/build: $O(nt)$ (Lemma E.2). Verify: $O(n^2 t)$ time, $O(n)$ space (Lemma E.8). Paths: T_{paths} time with $O(nt + L + J)$ (enum) or $O(\ell_{\max})$ (sample) auxiliary space, absorbed into $O(nt)$. Scoring/aggregation: $O(N \cdot \bar{p})$. Summing gives the claims. □

E.4 The Mostly-Linear Regime

The deduplication policy does not change the per-statement worst case (each constraint is still evaluated per step). It, however, bounds total emitted events at $O(|\mathcal{K}|) = O(n)$ rather than $O(n^2)$. In practice constraint sets rarely reach their worst-case size. Thus, in this section we show that Type-6 verification can be made *mostly* linear. First, we define the following:

Definition E.13 (Mostly-linear graph)**.** We say a graph G is *mostly-linear* if $E = E_{\text{spine}} \sqcup E_{\text{par}} \sqcup E_{\text{back}}$, where E_{spine} is a forward directed path (the *spine*) visiting every non-blank node at most once and covering all propositions in first-appearance order; E_{par} are parallel-edge duplicates of spine/par edges; and E_{back} are *backward* edges (target precedes source in spine order). We write $B = |E_{\text{back}}|$ (so $B \geq J$, with equality when every backward edge is a revisit edge).

Thus, in this section we prove the following:

Theorem E.14 (Mostly-linear total)**.** *For a mostly-linear graph from n statements with $t, \mu, \Delta, L, \kappa_{\max} = O(1)$ and $B = O(\log n)$,*

$$T_{\text{total}} = \begin{cases} O(n + \min(K, n^c) \cdot n) & \textit{(enumeration),} \\ O(n + S \cdot n^2) & \textit{(sampling),} \end{cases}$$

for a constant c depending on backward-edge sparsity, in $O(n + N \cdot \bar{p})$ space.

Observe that the spine order in a mostly-linear graph is exactly the jump and loopback edges, plus any Ω_b-labelled edge returning to a prior proposition. Hence, the decomposition from Definition E.13 reflects construction: each statement extends the spine forward by $O(t)$ edges; parallel edges arise when a statement lands on an existing proposition via a new operator (e.g., AND a on a prior a); backward edges arise only from R-chain closures (loopback), revisit prefixes (jump), or connective edges to an earlier proposition. Gap edges are always forward.

We now introduce two technical lemmas for general mostly-linear multi-digraphs.

Lemma E.15 (Path count, mostly-linear)**.** *The number of edge-simple source-to-sink paths in a mostly-linear multi-digraph, respecting the three policies, is $N_{\text{total}} = O(\mu^{|V|} \cdot 2^B)$.*

Proof. Fix any edge-simple path π. Being node-simple away from the sink, and with the spine in fixed order, the spine nodes π touches form a monotone subsequence determined by the backward edges π traverses: each backward edge selects a jump-back point, and between traversals the walk runs strictly forward along a contiguous segment. Thus π is fixed by (i) a subset $S \subseteq E_{\text{back}}$ (ordering forced up to constants) and (ii) a choice of $\leq \mu$ parallel edges per transition. Bounding (i) by 2^B and (ii) by $\mu^{|V|}$ (under $L + J = O(|V|)$) gives the claim. □

Lemma E.16 (Spine length)**.** *In a mostly-linear graph from n statements with per-statement bound t, $|E_{\text{spine}}| = O(nt)$ and $|V| = O(nt)$.*

Proof. Each statement contributes $O(t)$ nodes and edges (Lemma E.2); the spine is a subset of forward edges, one per added node, so $|E_{\text{spine}}| \leq |V| - 1 = O(nt)$. □

With both we can now show that EnumeratePaths and SamplePaths in the mostly-linear regime runs in polynomial (mostly linear!) time:

Lemma E.17 (Enumeration, mostly-linear)**.** *For a mostly-linear graph, EnumeratePaths with cap K runs in*

$$T_{\text{enum}} = O\Big(\min(K, \mu^{|V|} \cdot 2^B) \cdot (|V| + L + J) \cdot \Delta + W_{\text{dead}}\Big).$$

When $B = O(\log |V|)$ and $\mu = O(1)$ (the typical case) the path count is polynomial in $|V|$ and enumeration runs in polynomial time.

Proof. Substitute Lemma E.15 into Lemma E.3. Under these assumptions $\mu^{|V|} \cdot 2^B = O(|V|^c)$, so the leading term is polynomial in $|V|$, hence in n by Lemma E.16. □

Lemma E.18 (Sampling, mostly-linear)**.** *With $\ell_{\max} = O(|V| + L + J)$, SamplePaths runs in $T_{\text{sample}} = O(S \cdot (|V| + L + J)^2 \cdot \Delta)$. When $L = O(1)$, $J = O(\log n)$, $\Delta = O(1)$, this is $O(S \cdot n^2 t^2)$, or $O(S \cdot n^2)$ when $t = O(1)$.*

Proof. Straightforward, from substituting $\ell_{\max} = O(|V| + L + J)$ into Lemma E.6 and using Lemma E.16 for simplification. □

Lemma E.19 (Dead-end work, mostly-linear)**.** *For a mostly-linear graph, $W_{\text{dead}} = O((|V| + L + J) \cdot \Delta \cdot 2^B)$.*

Proof. A branch fails to yield only by exhausting feasible out-edges before the sink. Forward spine edges either advance or terminate at a blank ($O(\Delta)$ work, then return). Dead-end branches thus correspond to backward-edge choice sequences failing to re-enter the spine, $\leq 2^B$ of them; each has depth $O(|V| + L + J)$ (Lemma E.4(ii)) with $O(\Delta)$ per frame. □

Lemma E.20 (Verifier cost, mostly-linear)**.** *On a mostly-linear graph with $B = O(\log n)$ and $\kappa_{\max} = O(1)$, $T_{\text{verify}} = O(n)$.*

Proof. Per-statement update and chain work are $O(t) = O(1)$ (Proposition E.9), totalling $O(n)$. Back-propagation costs $O(|\mathcal{K}_i| \cdot \kappa_{\max}) = O(|\mathcal{K}_i|)$ (Proposition E.10); when prior constraints are largely inert. Also recall from earlier that the deduplication policy bounds total re-emitted events at $O(n)$. □

We are now ready to prove Theorem E.14:

Theorem E.21 (Mostly-linear total)**.** *For a mostly-linear graph from n statements with $t, \mu, \Delta, L, \kappa_{\max} = O(1)$ and $B = O(\log n)$,*

$$T_{\text{total}} = \begin{cases} O(n + \min(K, n^c) \cdot n) & \textit{(enumeration),} \\ O(n + S \cdot n^2) & \textit{(sampling),} \end{cases}$$

for a constant c depending on backward-edge sparsity, in $O(n + N \cdot \bar{p})$ space.

Proof. Parse/build $O(n)$ (Lemma E.2); verify $O(n)$ (Lemma E.20); paths $O(\min(K, n^c) \cdot n)$ (Lemma E.17, with W_{dead} absorbed by Lemma E.19) or $O(S \cdot n^2)$ (Lemma E.18); scoring $O(N \cdot \bar{p})$ with $N = O(\min(K, n^c))$ or S and $\bar{p} = O(n)$, absorbed into the leading term. Summing gives the claim. □

CoT traces of length $n = 20$–100 produce graphs with $|V| \approx n$, $\Delta \leq 4$, $\mu \leq 2$, and a handful of backward edges. Enumeration stays within the default cap $K = 10^4$, the coherence report is produced in time polynomial in trace length, and the verifier runs linearly. Hence, in the practical regime the pipeline is polynomial in trace length and tractable for realistic CoT sizes.

F Extended Results

F.1 Hard-fail and trace-level scores per model

Hard-fail rates broken down by model and dataset split are in Table 5. LINC's near-zero hard-fail rates (0-5% across all generators and splits) are a definitional consequence of its formal-verification mechanism: LINC hard-fails only when the LLM-generated FOL translation is syntactically ill-formed, not when the reasoning is substantively wrong. This is a **very different notion of 'failure'** from the LLM-judge or PRM methods, which flag reasoning-level errors, and motivates our reporting of both hard-fail agreement and scalar correlations rather than either alone. LLM-judge hard-fail rates cluster strongly by judge capability: capable judges (Claude Opus, GPT-5.6, Gemma-4B) hard-fail 50-70% of traces, while smaller or less-aligned judges (Qwen-3.5-9B, Llama-3.1-8B, GLM-4.7) hard-fail 97-100%. This spread suggests hard-fail as measured is at least as much a property of the LLM as of the traces themselves. Finally, the PRM methods we tested exhibit near-total hard-fail rates (90-100%) on natural-language reasoning traces despite being trained for step-wise correctness signals in mathematics; this is a **cross-domain transfer failure** worth further study.

In Table 6 we report mean trace-level scores over the same breakdown. As before, LLM-judges span a wide range of mean scores (2.0-4.2), and their rankings across generation models are not consistent. PRM methods produce mean scores from 0.22 to 4.32 depending on model, with the near-zero mean of Qwen-2.5-PRM reflecting its consistent near-boundary decisions. Type-6 sits between the LLM-judge and PRM regimes, at a mean of 1.34 overall with a notable spike on the MMLU split (2.06). The trivial baseline sits at 2.60 across all

Hard-fail rate	CoT-Logic	DebateLab	MMLU	Hand-crafted	All
LINC					
gemma4_e4b	0.0% (183)	0.0% (228)	0.0% (348)	0.0% (28)	0.0% (787)
gpt_56	1.1% (183)	7.0% (273)	3.8% (346)	17.6% (34)	4.8% (836)
qwen35_9b	0.0% (225)	0.0% (257)	0.0% (336)	0.0% (35)	0.0% (853)
LLM-as-a-judge					
opus48	69.6% (276)	84.2% (311)	51.4% (370)	84.6% (39)	68.0% (996)
gemma4	62.1% (256)	54.2% (299)	66.3% (362)	64.9% (37)	61.3% (954)
glm47	100.0% (148)	100.0% (186)	99.3% (150)	100.0% (16)	99.8% (500)
gpt56	63.1% (274)	79.4% (311)	50.5% (370)	76.9% (39)	64.1% (994)
llama31	98.0% (100)	99.0% (103)	97.5% (161)	100.0% (15)	98.2% (379)
qwen35	99.6% (227)	100.0% (256)	95.4% (304)	94.6% (37)	97.9% (824)
PRM					
llama31-prm	64.9% (276)	73.0% (311)	64.6% (370)	92.3% (39)	68.4% (996)
math-shepherd	98.6% (276)	100.0% (311)	90.8% (370)	100.0% (39)	96.2% (996)
qwen25-prm	100.0% (276)	100.0% (311)	100.0% (370)	100.0% (39)	100.0% (996)
ROSCOE					
deberta-large	100.0% (276)	100.0% (311)	100.0% (370)	100.0% (39)	100.0% (996)
Trivial					
trivial	0.0% (276)	0.6% (311)	0.0% (370)	0.0% (39)	0.2% (996)
Type-6					
type6	87.9% (273)	89.0% (309)	79.2% (370)	94.7% (38)	85.3% (990)

Table 5: Hard-fail rates by generation model and evaluation split, per method. Each cell reports the fraction of traces in the split that the method flagged as failing, with the number of traces in parentheses. Higher rates indicate the method rejects more traces; they do not directly indicate accuracy, as ground truth in most splits is unknown. LINC (three models, top block) produces a formal verdict on translated first-order-logic hypotheses and rejects a trace only when the formal check itself fails, giving very low hard-fail rates by construction. Trivial (bottom block) is a uniform-random baseline calibrated to reject ∼0% of traces. Between these extremes, LLM-judge, PRM, and Type-6 methods vary substantially across generation model and split, reflecting both genuine reasoning-quality variation across generators and each method's internal sensitivity to trace surface features.

splits by construction, providing a fixed reference line: any method whose mean score is close to this line is not distinguishing high- from low-quality traces on average, though it may still be doing so per-trace.

F.2 Per-model analysis

We discuss further details from the ablation study from Section **??**.

Trivial baseline. Trivial's four signals span the corpus with median statement count 55.5 (IQR $[41, 75]$) and typ–token ratio median 0.376. The non-trivial correlation with LLM-judges does not necessarily indicate false positives on the judges' part: shorter or more repetitive traces may be genuinely less coherent, and Trivial may be picking up a real signal. Table 7 reports the distribution of the four Trivial signals. Ranges are broad on all four, providing informative surface features despite the baseline's by-construction low hard-fail rate.

F.2.1 ROSCOE

ROSCOE. Individual ROSCOE metrics carry informative distributions: chain self-consistency median 0.056 (IQR $[0.042, 0.070]$); CSE-step (contradiction) median 0.033 (IQR $[0.018, 0.063]$). The degenerate hard-fail flag is therefore an artefact of ROSCOE's default threshold rather than of its underlying signals; the threshold was calibrated on datasets other than natural-language reasoning and does not transfer here. Table 8 reports the dis-

Mean trace score	CoT-Logic	DebateLab	MMLU	Hand-crafted	All
linc:gemma4_e4b	—	—	—	—	—
linc:gpt_56	—	—	—	—	—
linc:qwen35_9b	—	—	—	—	—
llm_judge:claude_opus_48	3.446	3.068	3.741	3.026	3.421
llm_judge:gemma4_e4b	4.277	4.174	4.022	4.108	4.142
llm_judge:glm_47	3.209	3.081	2.713	2.625	2.994
llm_judge:gpt_56	3.507	3.048	3.711	3.026	3.421
llm_judge:llama_31_8b	3.640	3.379	2.957	3.400	3.269
llm_judge:qwen35_9b	2.004	1.941	2.132	2.000	2.032
prm:llama31-8b-prm	4.374	4.249	4.367	4.183	4.325
prm:math-shepherd	1.724	1.470	2.145	1.602	1.796
prm:qwen25-prm	0.208	0.207	0.226	0.224	0.215
roscoe:deberta-large	1.414	1.423	1.199	1.381	1.336
trivial:trivial	2.593	2.648	2.559	2.563	2.596
type6	0.850	0.976	2.062	0.855	1.342

Table 6: Mean trace-level coherence score $\in [0, 5]$ by generation model and split, per method. Each cell reports the method-emitted or method-derived scalar averaged across traces in the split. LINC rows are marked with '—' because LINC produces a binary verdict rather than a scalar coherence score, and no principled projection to $[0, 5]$ exists; agreement between LINC and the scalar methods is reported through hard-fail agreement (Table **??**) instead.

Signal	Min	Q1	Median	Q3	Max
Statement count	4	41.0	55.5	75.0	326
Connective density	0.000	0.183	0.247	0.316	0.772
Repetition rate	0.000	0.022	0.046	0.089	0.386
Type-token ratio	0.080	0.301	0.376	0.437	0.740

Table 7: Distributional summary of the four Trivial signals.

tribution of the six ROSCOE metrics used to compute the ROSCOE trace-level scalar and hard-fail flag.

LINC. Table 9 reports corpus-wide LINC translation-fidelity distributions and verdict counts. Predicate grounding is at 1.000 across all three LINC models, suggesting that ontology extraction is a stable operation even when the source-to-logic translation is not. Table **??** disaggregate these by underlying translation model. As discussed in the main text, only GPT-5.6 produces informative entailment/non-entailment verdicts on this corpus; Qwen-3.5-9B and GLM-4.7 emit unknown or error verdicts on nearly all traces despite Qwen achieving the highest coverage ratio of the three. All models showed zero within-method agreement (omitted). The observation that GPT-5.6 achieves the lowest coverage ratio but the only informative verdict distribution suggests that when LINC's translation is confident enough to complete, it is also confident enough to commit to a verdict; when translation is partial, LINC reserves judgement and emits 'unknown'.

PRMs. The progressive collapse of 'pass' emissions from Llama (68% hard-fail) through Math-Shepherd (96%) to Qwen (100%) suggests that transfer to natural-language reasoning degrades sharply with the specificity of the PRM's mathematical training distribution. Math-Shepherd's non-zero low-confidence rate (5.3%) provides a modest reference point: when the PRM has any signal of uncertainty at all, it produces it; when it does not, as with Llama and Qwen at 0% low-confidence, mislabelling is confident. Table 12 reports the corpus-wide distribution of per-trace PRM aggregations, and Table 13 disaggregates the operational signals by PRM model. Table **??** confirms within-method disagreement.

Metric	Min	Q1	Median	Q3	Max
Faithfulness-step	0.037	0.147	0.264	0.360	0.538
Informativeness-step	0.163	0.251	0.284	0.322	0.508
Repetition-step	0.281	0.559	0.600	0.651	0.863
Reasoning alignment	0.040	0.299	0.397	0.510	0.837
Chain self-consistency	0.001	0.042	0.056	0.070	0.313
CSE-step (contradiction)	0.002	0.018	0.033	0.063	0.432

Table 8: Distributional summary of ROSCOE metrics.

Fidelity metric	Min	Q1	Median	Q3	Max
Coverage ratio	0.001	0.221	0.329	0.431	1.000
Predicate grounding	0.000	1.000	1.000	1.000	1.000
Parse errors per trace	0	0.0	1.0	4.0	61
Undeclared predicates	0	0.0	0.0	0.0	60
Verdict	**Count**	**% of corpus**			
Entailment	38	3.8%			
Non-entailment	189	19.0%			
Contradiction (hard-fail)	14	1.4%			
Unknown	746	74.9%			
Error (parse, timeout, or translation failure)	9	0.9%			

Table 9: LINC summaries of translation fidelity (top) and verdict distributions (bottom). Coverage ratio is the fraction of source premises translated; predicate grounding the fraction of predicates in the trace-extracted ontology. For verdicts, the hard-fail flag fires on contradiction.

LLMs-as-judges. Per-criterion flag rates reveal the underlying instability of LLM-judge measurements. 'Unresolved doubt' is flagged in 52-87% of traces depending on judge, and 'modal mismatch' in 29-95%; no single criterion produces stable rates across judges. This confirms that the judge-model effect is not narrowly attributable to overall scoring calibration but persists criteria by criteria in the rubric. Table 14 reports the corpus-wide flag rate for each of the four rubric axes (contradiction, unsupported conclusion, modal mismatch, unresolved doubt). Table 15 disaggregates these by judge model. Even on this per-criterion rubric, judges vary by two to three times in how often they flag a given criterion on the same corpus. Table 16 shows the resulting within-method pairwise agreement, with the maximum below $\kappa = 0.4$.

Type-6. Type-6's event incidence table shows that soft contradictions (B-B conflicts, unjustified downgrades) and residuals at sink are common, but only elevate to hard failures when they touch the direct implication chain. This design decision is what allows Type-6 to report a low false-positive rate: most of the corpus is context or exposition, and a naive contradiction-counter would hard-fail nearly everything. Table 17 reports the incidence of each Type-6 event category across the corpus, with the number of traces triggered and average events per triggered trace. Table 18 reports the direct-implication-chain statistics used to elevate residual signals to hard-fails. Table 19 reports the Spearman correlations between three alternative graded-reduction strategies for the Type-6 scalar, all of which are strongly rank-correlated.

Model	Entailment	Non-entail.	Contradiction	Unknown	Error
GPT-5.6	14.7%	65.3%	4.0%	0.0%	16.1%
Gemma-4-E4B	0.0%	0.0%	0.0%	79.0%	21.0%
Qwen-3.5-9B	0.0%	0.0%	0.0%	85.6%	14.4%
GLM-4.7	0.0%	0.0%	0.0%	72.8%	27.2%

Table 10: LINC verdicts by underlying translation model. Only GPT-5.6 produces informative entailment/non-entailment verdicts; smaller models emit predominantly unknown or error.

Model	Min	Q1	Median	Q3	Max
Coverage ratio					
GPT-5.6	0.000	0.004	0.006	0.008	0.018
Qwen-3.5-9B	0.033	0.328	0.515	0.714	1.024
GLM-4.7	0.001	0.006	0.010	0.014	0.049
Gemma-4-E4B	0.034	0.311	0.469	0.614	1.000
Parse errors per trace					
GPT-5.6	0	1.0	2.0	4.0	38
Qwen-3.5-9B	0	0.0	0.0	1.0	61
GLM-4.7	0	0.0	0.0	0.0	161
Gemma-4-E4B	0	0.0	0.0	3.0	36

Table 11: LINC translation fidelity by underlying model. Coverage ratio counts fraction of source premises formalised; higher is better. GPT-5.6 emits the most parse errors but achieves the lowest coverage ratio, indicating it fails on hard-to-translate premises rather than surface parsing.

Signal	Min	Q1	Median	Q3	Max
Trace mean score	0.267	0.392	0.426	0.453	0.564
Trace minimum score	0.005	0.037	0.067	0.197	0.454
Trace variance	0.0009	0.0087	0.0269	0.0404	0.0754
Signal	Traces	% of corpus			
Low-confidence (var < 0.01, mean $\in [0.4, 0.6]$)	0	0.0%			
Truncated (steps did not fit context)	39	3.9%			
Hard-fail (mean < 0.5 or min < 0.2)	967	97.1%			

Table 12: Distributional summary of PRM per-trace aggregations (top) and operational signals (bottom).

PRM	Low-confidence	Truncated	Hard-fail
Llama-3.1-8B-PRM	0.0%	2.7%	68.4%
Math-Shepherd	5.3%	3.6%	96.2%
Qwen-2.5-PRM	0.0%	4.0%	100.0%

Table 13: PRM operational signals by model. Low-confidence signals low-variance predictions near the boundary (var < 0.01, mean $\in [0.4, 0.6]$); hard-fail uses mean < 0.5 or minimum < 0.2.

Axis	Flag rate	κ with Type-6 analog
Contradiction	33.7%	—
Unsupported conclusion	62.0%	—
Modal mismatch	60.7%	—
Unresolved doubt	80.3%	—

Table 14: LLM-judge per-axis flag rates and agreement with the analogous Type-6 category: contradiction $\leftrightarrow$ K-K plus derived; unsupported conclusion $\leftrightarrow$ unverifiable derivation plus reasoning-avoidance; modal mismatch $\leftrightarrow$ modal mismatch (U_k/K); unresolved doubt $\leftrightarrow$ residual unresolved-doubt at sink.

Judge	Contradict.	Unsup. concl.	Modal mismatch	Unresolved doubt
Claude Opus 4.8	23.7%	54.7%	29.2%	66.3%
Gemma-4-E4B	2.6%	28.8%	49.8%	59.3%
GLM-4.7	45.2%	99.4%	38.8%	52.6%
GPT-5.6	34.3%	34.2%	43.3%	87.1%
Llama-3.1-8B	33.5%	38.0%	95.0%	72.8%
Qwen-3.5-9B	73.9%	95.9%	89.4%	85.2%

Table 15: LLM-judge per-axis flag rates by judge model. Rates vary by two to three times across judges on the same rubric axis and the same corpus.

	Claude Opus	Gemma-4	GLM-4.7	GPT-5.6	Llama-3.1	Qwen-3.5
Claude Opus	—					
Gemma-4	0.189	—				
GLM-4.7	0.008	0.006	—			
GPT-5.6	0.382	0.280	0.007	—		
Llama-3.1	−0.010	0.011	0.000	0.011	—	
Qwen-3.5	0.084	0.071	0.196	0.074	0.090	—

Table 16: Within-LLM-judge pairwise Cohen's κ on hard-fail. Even the highest agreement ($\kappa = 0.382$ between Claude Opus and GPT-5.6) is below moderate.

Category	Traces triggered	Avg. events/triggered	Blocks path?
Hard contradictions			
K-K contradiction	62	2.258	✓
Derived contradiction	409	5.276	✓
Modal mismatch (U_c)	36	1.417	✓
Modal mismatch (U_k/K)	182	2.308	✓
Reasoning-avoidance	548	—	✓(trace)
Soft contradictions			
B-B conflict	87	2.805	
Bare-reassertion conflict	68	2.206	
Unjustified downgrade	200	2.225	
Unjustified modal shift	75	1.680	
Residual (at sink)			
Unresolved doubt	495	—	
Unresolved unknowability	878	—	
Quality signals			
Redundant re-assertion	171	2.000	
Self-questioned K	0	—	
Self-questioned B	14	2.000	
Unverifiable derivation	250	2.716	
Ambiguous negated connective	72	2.792	

Table 17: Incidence of each Type-6 event category. 'Traces triggered' counts traces with at least one event of that category; the average is the mean event count within those traces. Residual signals at sink are elevated to hard when on the direct implication chain (Section 3).

Statistic	Min	Q1	Median	Q3	Max
Chain length (props on direct implication chain to sink)	1	1.0	1.0	3.0	32
On-chain fraction (chain length / trace length)	0.003	0.016	0.025	0.074	0.733
Elevated residuals per trace	0	0.0	0.0	0.0	16

Table 18: Direct-implication-chain statistics. Higher on-chain fractions indicate tighter reasoning; elevated residuals count the unresolved-doubt or unresolved-U_k signals raised to hard by touching the chain.

	strict	graded	proportional
strict	—		
graded	0.703	—	
proportional	0.697	0.976	—

Table 19: Spearman ρ between Type-6 alternative reduction strategies on the trace-level coherence scalar. All three are strongly rank-correlated, indicating the choice of reduction is not a large source of variance in Type-6's output.