

ABED: Assumption-Based Bidirectional Entailment Diagnosis for Open-World Logical Reasoning

Seyeon Jeong , Sojeong Park , Beakcheol Jang*

Graduate School of Information, Yonsei University, Seoul, South Korea

{seyeonjeong, qkrthwj714, bjang}@yonsei.ac.kr

Abstract

Open-world logical entailment requires deciding whether a set of premises entails a conclusion, entails its negation, or leaves it undecided. Many existing methods are deductive in spirit, searching for a proof and reading the label from whether that proof succeeds. When implemented through LLM-based proof search, this makes the prediction depend on a single unstable trajectory, and a failed or partial search is easily mismatched to the wrong label. We propose ABED (Assumption-Based Bidirectional Entailment Diagnosis), a training-free inference-time framework that reframes entailment classification as missing assumption diagnosis. Instead of asking only whether a conclusion can be proved, ABED asks what assumption is missing for it to hold, and classifies that gap as already closed by the premises (Null), contradicted by them (Contra), or left open (Admissible). It applies this diagnosis to both a conclusion and its negation, and determines the label from the resulting dual gap pair. Across FOLIO, ProofWriter, Pron-toQA, and LogicNLI, ABED consistently outperforms prompting-based, symbolic, solver-based, and deductive baselines, improving over the strongest baseline on each dataset by 3.4, 5.3, 0.8, and 4.4 points with GPT-4.1.

1 Introduction

Large language models have achieved strong performance across many reasoning tasks (Brown et al., 2020; Ouyang et al., 2022; Patel et al., 2022; Guo et al., 2025; Yang et al., 2025), yet logical reasoning under open-world semantics remains challenging (Wei et al., 2022; Cheng et al., 2025; Liu et al., 2025a). Many existing approaches to logical entailment are deductive in spirit. They derive a conclusion from the premises, or seek evidence against it, and assign a label from the resulting proof process (Pan et al., 2023; Xu et al., 2024,

*Corresponding author.

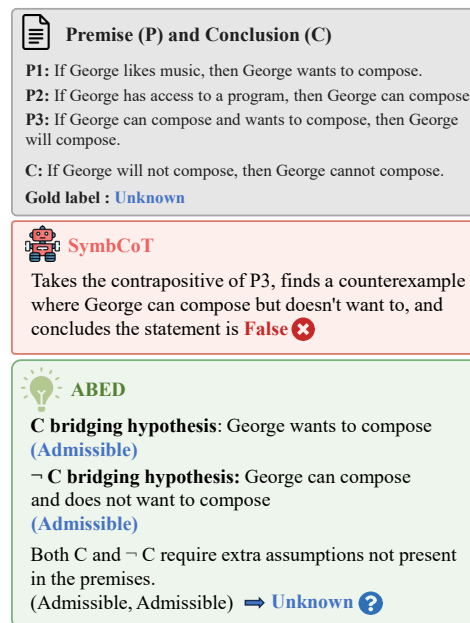


Figure 1: Motivating example where a proof-search baseline over-reads a counterexample as *False*, while ABED identifies open gaps for both *C* and $\neg C$ and predicts *Unknown*.

2025; Tian et al., 2021). This is natural, since entailment is traditionally defined through derivability. However, when deduction is implemented through LLM-based proof search, predictions can be affected by proof-search failures, incomplete reasoning, and overconfident interpretations of partial evidence (Saparov and He, 2022; Pan et al., 2023; Xu et al., 2024; Song et al., 2025). Under open-world semantics, where a conclusion may be supported, contradicted, or left undecided, inference should diagnose what the premises establish, contradict, or leave open, rather than rely only on a single proof trace (Han et al., 2024; Tafjord et al., 2021).

Figure 1 illustrates this failure mode. Given the premise “If George can compose and wants to compose, then George will compose” and the conclu-

sion “If George will not compose, then George cannot compose,” a proof-search model may find a counterexample where George can compose but does not want to compose. This only shows that the conclusion is not guaranteed; it does not show that the premises entail its negation, so the correct label is *Unknown*. ABED targets this mapping error by diagnosing what is missing for both C and $\neg C$, rather than trusting a single proof trajectory.

To reduce this dependence on unstable proof-search outcomes, we diagnose the missing assumption behind a conclusion. When a conclusion is not immediately established, we ask what additional assumption would make it follow, and whether that assumption is supported, contradicted, or left open by the premises. This turns an open-ended proof search into a bounded check on a single named proposition. The assumption itself is never returned as an answer or treated as evidence. It exists only to make the gap explicit enough to verify. Related ideas appear in NLP work on explanation selection and missing-fact generation (Bhagavatula et al., 2019; Tafjord et al., 2021), where generating a missing hypothesis is the task itself. In ABED it is instead an internal diagnostic step, and deductive checks determine how the diagnosed gap relates to the premises.

We propose ABED (Assumption-Based Bidirectional Entailment Diagnosis), a training-free inference-time framework for missing assumption diagnosis. For a target conclusion, ABED reasons backward to a candidate bridging hypothesis and classifies the gap as already closed by the premises (*Null*), contradicted by them (*Contra*), or neither (*Admissible*). It applies this diagnosis to both the conclusion and its negation, mapping the resulting dual gap pair to *True*, *False*, or *Unknown*. Labels are thus read from verified gap states rather than from whether a single proof of C succeeds, unlike deductively grounded methods (Xu et al., 2024, 2025); and unlike prior NLP work that treats missing-hypothesis generation as a standalone task (Bhagavatula et al., 2019; Tafjord et al., 2021), ABED uses it as an internal diagnostic signal within entailment classification.

Returning to the example, ABED finds that deriving the conclusion requires the assumption “George wants to compose.” This assumption is neither stated nor contradicted, so it forms an *Admissible* gap. Establishing the opposite conclusion requires that George can compose and does not want to compose. This is also neither stated nor contradicted.

Since both gaps remain open, ABED predicts *Unknown* because the underdetermination is explicitly diagnosed.

ABED consists of four stages: assumption gap estimation, boolean gap verification, consistency-constrained dual alignment, and a restricted symbolic audit for unresolved admissible cases.

Across four logical reasoning benchmarks, ABED achieves 80.4 on FOLIO (Han et al., 2024), 95.5 on ProofWriter (Tafjord et al., 2021), 100.0 on ProntoQA (Saparov and He, 2022), and 74.7 on LogicNLI (Tian et al., 2021) with GPT-4.1, consistently outperforming prompting-based, symbolic, solver-based, and deductive reasoning baselines.

Our contributions are as follows.

- We reframe open-world logical entailment classification as missing assumption diagnosis.
- We propose ABED, a training-free inference-time framework that diagnoses both a conclusion and its negation using *Null*, *Contra*, and *Admissible* gap states, combining assumption gap estimation, boolean verification, consistency alignment, and a localized symbolic audit.
- We show that ABED consistently improves over prompting-based, symbolic, solver-based, and deductive baselines across four logical reasoning benchmarks.

2 Related Work

Large language models still show systematic weaknesses in logical reasoning, particularly on tasks involving multi-step inference, conditionals, or incomplete premises.

2.1 Prompt-Based Logical Reasoning

Prompt-based methods guide LLM reasoning through structured prompts without modifying parameters. Chain-of-Thought prompting (Wei et al., 2022) elicits intermediate reasoning steps before the final answer, and self-consistency (Wang et al., 2022) samples multiple reasoning paths and aggregates them by majority vote. As concerns grew that natural-language CoT may not preserve logical structure reliably, later work injected symbolic representations into the reasoning process. SymbCoT (Xu et al., 2024) translates problems into symbolic logic before stepwise reasoning, LoT (Liu et al., 2025b) adds propositional logic expressions to the

input context, and Aristotle (Xu et al., 2025) internalizes formal reasoning patterns and introduces an entailment classifier based on resolution and proof by contradiction. These methods make reasoning more explicit, but the final judgment still rests largely on the model’s own generated proof. Maieutic Prompting takes a different route within prompting, recursively generating a tree of explanations for both a proposition and its negation and resolving them as a satisfiability problem over their logical relations (Jung et al., 2022). Consistency there is a constraint on generation rather than a diagnosed state.

2.2 Solver-Based Logical Reasoning

Solver-based methods translate natural-language problems into formal representations such as first-order logic, logic programming, or SAT and SMT formulations, and then delegate inference to external engines. Logic-LM (Pan et al., 2023) passes LLM-translated symbolic forms to external solvers. LINC (Olausson et al., 2023) aggregates multiple symbolic translations to reduce translation errors. More recent work decomposes complex premises into verifiable sub-formulas (Ryu et al., 2025) or combines LLMs with reasoning engines in a solver-agnostic manner (Callewaert et al., 2025). These methods offer formal verifiability, but depend on complete and accurate symbolic translation. Errors introduced during translation propagate through the pipeline, especially when premises contain implicit or underspecified relations that are hard to formalize. SymBa instead lets a symbolic solver control the proof process, calling the LLM only when the solver needs new information, which yields complete SLD-resolution-style backward chaining over the recovered theory (Lee and Hwang, 2025). ABED shares the goal of limiting what the LLM decides, but applies it at a different granularity. The LLM proposes a single named proposition whose status is then checked, rather than supplying premises to a solver-driven search.

2.3 Missing-Hypothesis Generation for Incomplete Information

A long line of work infers a plausible missing condition from an observed outcome, which is useful when information is incomplete (Denecker and Kakas, 2002; Abe et al., 2024; Salimi et al., 2026). In NLP, the α NLI benchmark casts this as selecting the most plausible explanation for a narrative transition (Bhagavatula et al., 2019), and later work

exploits mutual exclusivity among candidate explanations to distinguish plausible ones from merely fluent ones (Zhao et al., 2023). ProofWriter includes a mode that generates missing facts for unproven conclusions (Tafjord et al., 2021), and other work applies similar ideas to filtering irrelevant information (He and Lu, 2024) or backward-chaining search from the conclusion (Kazemi et al., 2023). In these settings the generated hypothesis is the answer. ABED uses the same operation for a different purpose. The bridge it generates is an intermediate object, and the label is read from how that bridge stands against the premises.

3 Method

We propose **ABED**, a missing assumption diagnosis framework for open-world logical entailment. Standard deductive methods ask directly whether a conclusion C follows from a premise set P , and read the answer from whether that proof succeeds. ABED instead asks a reverse-direction question: *what is missing for a target hypothesis to follow from the premises?* It applies this question symmetrically to both C and its negation $\neg C$, and compares the two diagnosed gaps to decide whether the answer is *True*, *False*, or *Unknown*.

Realizing this idea requires four steps. **Assumption Gap Estimation** (AGE) proposes a compact missing assumption for each target. **Boolean Gap Verification** (BGV) checks this proposal through constrained yes/no entailment questions. **Consistency-Constrained Dual Alignment** (CCDA) aligns the two independently verified directions into a valid dual-pair decision. Finally, **Admissibility Verification** applies restricted Symbolic Forward Chaining (SFC) only to unresolved admissible pairs.

3.1 Problem Setup and Logical Preliminaries

Given a set of natural-language premises $P = \{p_1, \dots, p_n\}$ and a conclusion C , the task is to predict one of three labels, $A \in \{True, False, Unknown\}$. Following the derivability-based formulation in Aristotle (Xu et al., 2025), we define the labels as in Equation (1).

$$\begin{aligned} True &\iff P \vdash C, \\ False &\iff P \vdash \neg C, \\ Unknown &\iff P \not\vdash C \wedge P \not\vdash \neg C. \end{aligned} \tag{1}$$

Here, $P \vdash C$ denotes that C is derivable from P . We assume P is internally consistent unless

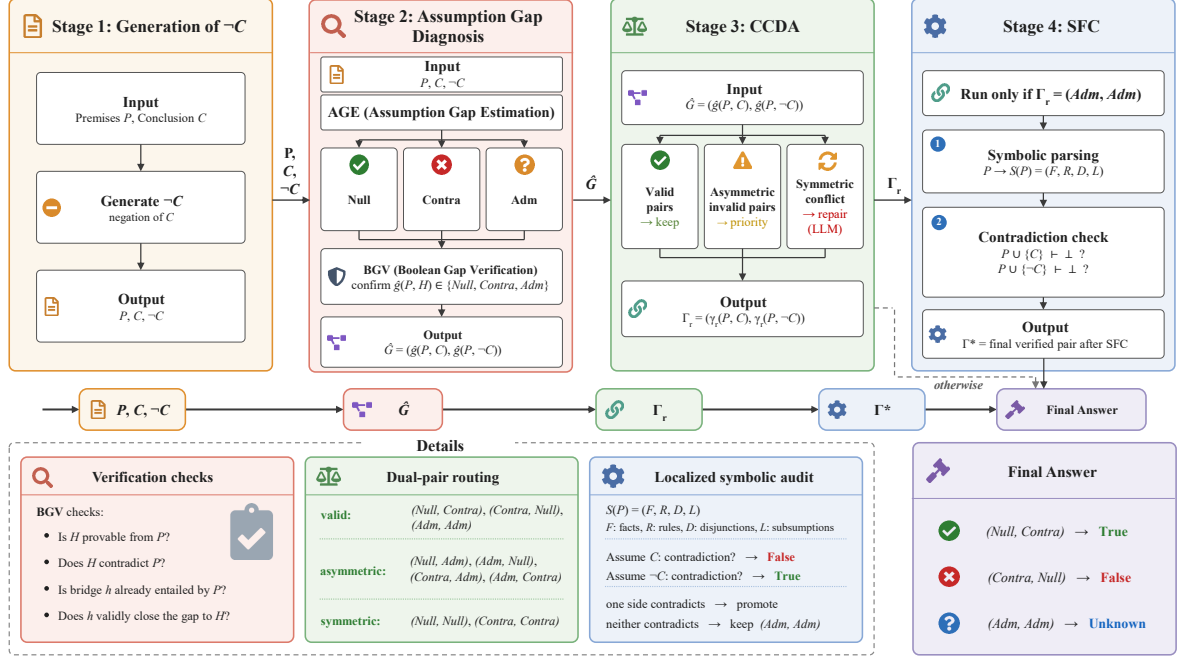


Figure 2: Overview of ABED. Given premises P and a conclusion C , Stage 1 generates the strict negation $\neg C$ to form the two targets C and $\neg C$. Stage 2 performs assumption gap diagnosis with AGE and BGV, producing verified gap states for both directions. Stage 3 applies CCDA to route the resulting dual gap pair, keeping valid pairs, repairing incomplete asymmetric pairs by determinate priority, and rechecking symmetric conflicts. Stage 4 invokes SFC only for unresolved (Adm, Adm) cases, using a localized symbolic audit to recover missed forward proofs. The final answer is obtained by mapping the verified dual pair to *True*, *False*, or *Unknown*.

stated otherwise. For binary yes/no benchmarks, we output *yes* only when $P \vdash C$, and map both *False* and *Unknown* to *no*.

Rather than assigning a label from a single proof attempt, ABED characterizes the gap between P and a target hypothesis $H \in \{C, \neg C\}$ through a gap state $\gamma(P, H)$, defined in Equation (2).

$$\gamma(P, H) = \begin{cases} \text{Null}, & \text{if } P \vdash H, \\ \text{Contra}, & \text{if } P \cup \{H\} \vdash \perp, \\ \text{Adm}, & \text{otherwise.} \end{cases} \quad (2)$$

Null means H already follows from the premises, and *Contra* means assuming H contradicts them. *Adm* means H is compatible with the premises but not derivable from them; it is therefore not evidence for H , but a signal that the premises leave H open.

Applying this diagnosis to both C and $\neg C$ yields the dual gap pair $\Gamma(P, C) = (\gamma(P, C), \gamma(P, \neg C))$. When P is consistent, the valid pairs map to labels as in Equation (3).

$$\begin{aligned} (\text{Null}, \text{Contra}) &\rightarrow \text{True}, \\ (\text{Contra}, \text{Null}) &\rightarrow \text{False}, \\ (\text{Adm}, \text{Adm}) &\rightarrow \text{Unknown}. \end{aligned} \quad (3)$$

The goal of ABED is therefore to obtain a reliable dual gap pair for C and $\neg C$. Equations (2)–(3) define ideal semantic gap states, but in practice ABED observes estimated states produced by LLM-based diagnosis. Asymmetric pairs such as $(\text{Null}, \text{Adm})$ or $(\text{Contra}, \text{Adm})$ should therefore not be read as valid semantic states, but as under-verified outputs that the subsequent stages are designed to resolve.

3.2 Stage 1: Generation of $\neg C$

ABED first generates the proposition-level negation $\neg C$ of the conclusion, making the complementary target explicit for later comparison, following proof-by-contradiction formulations in prior work (Bishop, 1967; Xu et al., 2025). A constrained LLM prompt produces a strict semantic negation rather than a surface “not” prefix, handling universal, existential, and conditional forms while preserving entity names. The output is the shared tuple $(P, C, \neg C)$ used by all subsequent stages.

3.3 Stage 2: Assumption Gap Diagnosis

Stage 2 diagnoses the gap between the premises P and each target hypothesis $H \in \{C, \neg C\}$. It consists of two components. AGE first identifies

what is missing for H to follow from P , and BGV then verifies whether the proposed gap is genuine.

Given (P, H) , the Assumption Gap Estimator (AGE) reasons backward from the target hypothesis H . It outputs an initial gap state $\gamma_0(P, H)$ and a bridging hypothesis h , as shown in Equation (4).

$$\text{AGE}(P, H) \rightarrow (\gamma_0(P, H), h). \quad (4)$$

The bridging hypothesis h is not evidence for H . It is a diagnostic artifact that describes what additional assumption would be needed to make H derivable from the premises. AGE is constrained not to generate arbitrary explanations. The bridge should be compact, relevant to deriving H , and expressed using predicates or entities already present in the problem whenever possible. When multiple bridges are plausible, AGE is prompted to prefer the weakest bridge that closes the derivational gap without adding unrelated information. For example, if the premises state that a condition is sufficient for a result and also state that the result holds, the original condition does not follow from the premises alone. Deriving it would require an unsupported converse assumption. AGE exposes this missing assumption and therefore marks the case as admissible rather than entailed.

Because AGE relies on open-ended LLM generation, Boolean Gap Verification (BGV) verifies the proposed diagnosis with four boolean checks. BGV is not a second open-ended proof-generation step. Each check is formulated as a constrained binary entailment question with a fixed output schema, and the final state is assigned by a deterministic priority rule. The first check, b_1 , asks whether the target H is already derivable from P . The second check, b_2 , asks whether assuming H contradicts P . The third check, b_3 , asks whether the proposed bridge h is already supported by P . The fourth check, b_4 , asks whether adding h to P is sufficient to derive H without introducing a contradiction. Together, b_3 and b_4 detect pseudo-gaps, where AGE reports a missing assumption that is actually already licensed by the premises and sufficient to close the gap.

BGV assigns the verified gap state using the top-down priority rule in Equation (5). The rule is evaluated in order.

$$\hat{\gamma}(P, H) = \begin{cases} \text{Null}, & b_1, \\ \text{Contra}, & b_2, \\ \text{Null}, & b_3 \wedge b_4, \\ \text{Adm}, & \text{otherwise.} \end{cases} \quad (5)$$

The rule assigns *Null* when H is entailed (b_1) or when a pseudo-gap is detected ($b_3 \wedge b_4$), *Contra* when H contradicts P (b_2), and *Adm* otherwise. Applying AGE and BGV to both C and $\neg C$ produces the verified dual gap pair $\hat{\Gamma}(P, C) = (\hat{\gamma}(P, C), \hat{\gamma}(P, \neg C))$.

3.4 Stage 3: Dual-Pair Consistency Alignment

Because BGV verifies C and $\neg C$ separately, the resulting dual pair may still need alignment. It may be *incomplete*, with one direction determinate and the other *Adm*, or *logically invalid*, with both directions giving mutually incompatible determinate states. CCDA handles the former with a determinate-priority rule and the latter with an entailment re-judgment.

The determinate-priority rule rests on an asymmetry between the two kinds of states. *Null* and *Contra* are *positive* verification outcomes: BGV has actively confirmed derivability or incompatibility through its boolean checks. *Adm*, in contrast, is not a positive finding but a residual state, meaning only that neither derivability nor contradiction was established. These two kinds of signals are therefore not equally reliable, and this is not merely an assumption: as we report in Section 4.5 and Table 4, post-BGV determinate signals reach a precision of 0.92 to 1.00 across all datasets. When a determinate state and an *Adm* state disagree within a pair, it is thus far more likely that the *Adm* side is under-verified than that the high-precision determinate side is wrong. CCDA accordingly anchors the decision on the determinate side, treating the pair as under-verified rather than logically inconsistent.

This justifies the repair rule for incomplete asymmetric pairs in Equation (6).

$$\begin{aligned} (\text{Null}, \text{Adm}) &\rightarrow \text{True}, \\ (\text{Adm}, \text{Null}) &\rightarrow \text{False}, \\ (\text{Contra}, \text{Adm}) &\rightarrow \text{False}, \\ (\text{Adm}, \text{Contra}) &\rightarrow \text{True}. \end{aligned} \quad (6)$$

For example, $(\text{Null}, \text{Adm})$ is repaired to *True*: C has been positively verified as derivable, while the $\neg C$ direction only failed to reach a determinate state, so the verified side is taken as the decision.

The other three cases follow by the same reasoning, with *Contra* on a side acting as positive evidence against that target.

For logically invalid symmetric conflicts, the pairs $(Null, Null)$ and $(Contra, Contra)$ are impossible under a consistent premise set. Here neither side can be preferred on reliability grounds, since both are determinate, so CCDA does not apply the priority rule. Instead, it resolves the conflict with one additional LLM-based entailment re-judgment over C and $\neg C$: if only C is judged entailed, the pair becomes $(Null, Contra)$; if only $\neg C$ is judged entailed, it becomes $(Contra, Null)$; and if neither is confirmed, CCDA conservatively maps it to (Adm, Adm) . The output is the repaired pair $\Gamma_r(P, C) = (\gamma_r(P, C), \gamma_r(P, \neg C))$.

3.5 Stage 4: Symbolic Audit and Final Prediction

Stage 4 runs only when the repaired pair remains (Adm, Adm) . ABED does not immediately finalize this as *Unknown*, since the pair may reflect genuine underdetermination or a missed forward proof. To separate the two, ABED invokes Symbolic Forward Chaining (SFC) as a localized audit.

SFC is intentionally lightweight. Rather than translating the whole problem into a logic program or calling a general-purpose solver, it asks the LLM to extract a minimal symbolic representation $\mathcal{S}(P) = (\mathcal{F}, \mathcal{R}, \mathcal{D}, \mathcal{L})$, consisting of ground facts, simple implication rules, ground disjunctions, and predicate-level subsumption relations. Because this extraction is restricted to a small fragment and is invoked only for the few unresolved (Adm, Adm) cases, its translation surface is far smaller than that of solver-based pipelines, which limits the translation errors that such pipelines tend to propagate. A deterministic forward chainer then applies a small set of rules, including modus ponens, modus tollens, disjunctive syllogism, subsumption propagation, and case-split closure. SFC is thus a restricted audit, not a complete theorem prover.

SFC checks whether either side produces a one-sided contradiction. We write $OC(H)$ when SFC derives a contradiction from $P \cup \{H\}$ while the opposite assumption does not. The audited pair is given in Equation (7).

$$\Gamma^*(P, C) = \begin{cases} (Null, Contra) & \text{if } OC(\neg C), \\ (Contra, Null) & \text{if } OC(C), \\ (Adm, Adm) & \text{otherwise.} \end{cases} \quad (7)$$

$OC(\neg C)$ supports C , and $OC(C)$ supports $\neg C$; if both or neither side yields a contradiction, the pair stays (Adm, Adm) . The final prediction applies the mapping in Equation (3) to $\Gamma^*(P, C)$. SFC is triggered only after both directions remain admissible and can change the decision only on a one-sided contradiction, so it functions as a conservative audit against false *Unknown* predictions rather than as a general-purpose solver.

4 Experiments

4.1 Experimental Settings

LLMs We evaluate ABED and all baselines using GPT-4.1 (Achiam et al., 2023), GPT-4.1-mini (Achiam et al., 2023), and DeepSeek-V3.2 (Liu et al., 2024) as backbone models, spanning a strong proprietary model, a smaller variant, and a non-OpenAI model. Unless otherwise stated, all methods are evaluated under the same backbone for fair comparison.

Datasets We evaluate on four logical reasoning benchmarks. ProntoQA (Saparov and He, 2022) contains synthetic multi-step deductive chains evaluated as a binary yes/no task, while the other three use three-way *True/False/Unknown* labels. ProofWriter (Tafjord et al., 2021) consists of rule-based reasoning over synthetic facts and rules under the open-world assumption, with annotated reasoning depth. FOLIO (Han et al., 2024) is an expert-written first-order logic benchmark with more naturalistic premises, and LogicNLI (Tian et al., 2021) is an NLI-style first-order reasoning benchmark mapped to the same scheme. Further dataset details are provided in Appendix A.

Baselines We compare ABED with prompting-based, logic-oriented, and symbolic or solver-based baselines. The prompting baselines include Naive Prompting and Chain-of-Thought prompting (Wei et al., 2022). Logic-oriented baselines include DetermLR (Sun et al., 2024) and Aristotle (Xu et al., 2025). Symbolic and solver-based baselines include SymbCoT (Xu et al., 2024), Logic-LM (Pan et al., 2023), and LINC (Olausson et al., 2023). All baselines are re-run with the same backbone model as ABED whenever applicable, with further details in Appendix B.

Evaluation We report accuracy for all datasets. FOLIO, ProofWriter, and LogicNLI are evaluated as three-way classification over *True*, *False*, and

Method	GPT-4.1				GPT-4.1-mini				DeepSeekV3.2			
	FOLIO	PW	Pronto	Logic	FOLIO	PW	Pronto	Logic	FOLIO	PW	Pronto	Logic
Vanilla	70.1	60.8	76.0	68.0	69.2	53.3	78.2	65.0	68.1	71.0	61.6	63.3
CoT	67.7	60.5	92.0	68.3	75.9	77.2	90.0	67.0	68.3	72.0	58.8	65.0
LogicLM	76.0	73.3	96.4	65.8	73.5	68.5	95.3	68.3	73.7	83.6	82.2	64.8
SymbCoT	77.0	78.8	98.2	70.3	75.8	69.8	97.3	62.0	74.8	88.3	95.4	61.7
LINC	71.5	90.2	98.0	67.3	66.1	80.2	95.5	65.2	68.3	85.2	94.3	62.0
Aristotle	70.3	86.5	99.2	67.5	70.8	78.3	97.8	69.8	71.2	83.0	93.8	63.1
DetermLR	70.6	79.0	99.0	61.6	72.5	69.0	93.4	60.5	69.2	72.5	95.0	61.5
ABED	80.4	95.5	100.0	74.7	77.0	90.3	99.3	74.6	78.4	93.0	97.0	68.7

Table 1: Accuracy comparison across four logical reasoning benchmarks under three backbone models: GPT-4.1, GPT-4.1-mini, and DeepSeek-V3.2. PW denotes ProofWriter, Pronto denotes ProntoQA, and Logic denotes LogicNLI.

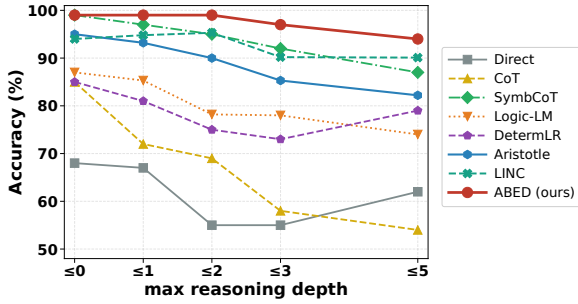


Figure 3: Accuracy by ProofWriter reasoning depth under GPT-4.1. ABED remains robust as depth increases, showing stronger stability on deeper multi-hop reasoning.

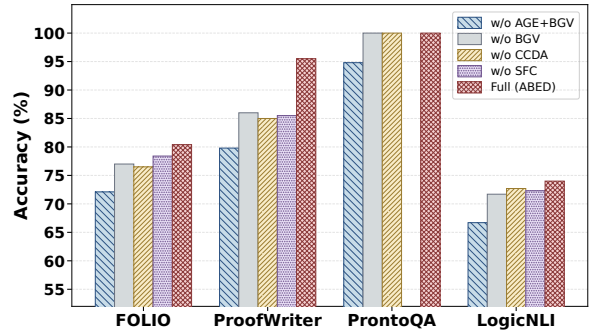


Figure 4: Ablation study of ABED on four benchmarks under GPT-4.1. The figure compares the full model with variants removing AGE+BGV, BGV, CCDA, and SFC.

Unknown, using the full ABED pipeline. ProntoQA is evaluated as a binary yes/no task. ABED applies the same gap diagnosis to the conclusion, but collapses non-entailed states into no. A Null state is mapped to yes, while Contra and Adm states are mapped to no.

4.2 Main Results

Table 1 reports the main accuracy results on the four benchmarks. ABED achieves the best accuracy on all four datasets under all three backbones, outperforming every prompting-based, logic-oriented, and solver-based baseline. With GPT-4.1, ABED improves over the strongest baseline on each dataset by 3.4 points on FOLIO (77.0 to 80.4), 5.3 on ProofWriter (90.2 to 95.5), 0.8 on ProntoQA (99.2 to 100.0), and 4.4 on LogicNLI (70.3 to 74.7). The margins are largest on FOLIO and ProofWriter, two open-world benchmarks that admit a genuine *Unknown* label, where high accuracy cannot be reached by determinate predictions alone. The gain on ProntoQA is by contrast small,

since the benchmark is close to saturation: it consists of synthetic, regularly structured deductive chains with binary labels, and the strongest baselines already score 98 to 99, so ABED reaching 100.0 reflects a near-saturated benchmark rather than an outlier. The same pattern holds under the other two backbones, with the largest gain on ProofWriter (10.1 points under GPT-4.1-mini), confirming that the gap-diagnosis mechanism is not tied to a single proprietary backbone. The improvement is most pronounced when the backbone is weaker and more likely to miss or incompletely track multi-step proof chains. Overall, these results support our main hypothesis: reasoning about what is missing between the premises and both C and $\neg C$ provides a stronger decision signal than directly asking the model for a final answer or a forward proof. ABED achieves these gains at a cost comparable to the strongest symbolic baseline; we report a full token and call-count analysis in Appendix D and pairwise statistical significance over all baselines in Appendix F.

Dataset	# Invalid	Inv→Unk.	+CCDA
FOLIO	36/204 (17.6%)	16.7	66.7
ProofWriter	70/600 (11.7%)	15.7	85.7
LogicNLI	60/300 (20.0%)	38.3	56.7

Table 2: Effect of CCDA under GPT-4.1 on invalid dual-gap pairs. *Unk.* denotes *Unknown*; directly mapping invalid pairs to *Unknown* discards many recoverable cases, while CCDA repairs them under valid-pair constraints.

4.3 Depth-wise Analysis on ProofWriter

To examine robustness under increasing reasoning complexity, we analyze ProofWriter accuracy by maximum reasoning depth. As shown in Figure 3, ABED stays at 97.0 or above through depth ≤ 3 and only declines to 94.0 at depth ≤ 5 , while most baselines degrade far more sharply: even the strongest, SymbCoT and Aristotle, fall to 87.0 and 82.2. As a result, ABED’s margin over every baseline widens with depth, from a tie at depth ≤ 0 to a 7.0-point lead over SymbCoT at depth ≤ 5 . This suggests that assumption gap diagnosis stabilizes multi-hop reasoning by explicitly checking what is missing for both C and $\neg C$, rather than relying on a single forward proof trajectory that becomes increasingly fragile as proof chains grow longer.

4.4 Ablation and Module Analysis

Figure 4 reports the ablation results with GPT-4.1. Full ABED achieves the strongest performance on FOLIO, ProofWriter, and LogicNLI, while ProntoQA remains nearly saturated across variants. The largest drops appear when AGE+BGV is removed, with accuracy falling by roughly 20–30 points on the three-way datasets, showing that assumption gap diagnosis is the core source of ABED’s improvement. Removing BGV, CCDA, or SFC also reduces performance, especially on ProofWriter, confirming that verification, dual-pair alignment, and symbolic audit further stabilize open-world multi-hop reasoning. We further analyze the two repair stages in Tables 2 and 3. Invalid dual-gap pairs are frequent, accounting for 11.7–20.0% of all instances across the three-way datasets. Both CCDA and SFC recover a substantial fraction of cases that direct *Unknown* assignment would lose, with the largest SFC gain on ProofWriter. ProntoQA varies little because it is evaluated in a binary setting where non-Null states are collapsed into no and most instances are already resolved at the diagnosis stage.

Dataset	Unk.	+SFC	Δ
FOLIO	62.5	69.0	+6.5
ProofWriter	72.5	96.1	+23.5
LogicNLI	75.0	78.0	+3.0

Table 3: Effect of SFC on symmetric *Adm/Adm* cases under GPT-4.1. *Unk.* directly maps unresolved pairs to *Unknown*, while +SFC applies the localized symbolic audit.

Dataset	Track	n	Prec.	Rec.
FOLIO	C	97	0.94	1.00
	$\neg C$	94	0.96	0.92
ProofWriter	C	297	0.99	1.00
	$\neg C$	306	1.00	0.99
LogicNLI	C	185	0.92	0.89
	$\neg C$	183	0.93	0.93
ProntoQA	$C/\neg C$	500	1.00	1.00

Table 4: Post-BGV single-target diagnosis quality under GPT-4.1. For each target, C or $\neg C$, we evaluate cases where BGV outputs a determinate state, i.e., *Null* or *Contra*. Precision and recall measure how reliably these determinate signals are identified before CCDA.

4.5 Single-target Diagnosis Quality

Table 4 evaluates whether BGV produces reliable determinate signals before dual-pair alignment. For each example, ABED runs BGV separately with the original conclusion C and with its negation $\neg C$ as the target. We then evaluate the cases where BGV outputs a determinate state, namely *Null* or *Contra*, for each target direction. Across datasets, these determinate signals show high precision: above 0.92 for both C and $\neg C$ on all three-way datasets, and perfect on ProntoQA. This supports CCDA’s determinate-priority repair. When one target direction is assigned a determinate state and the other remains *Adm*, CCDA can rely on the determinate side as the more reliable decision signal without making an additional LLM call.

4.6 Unknown Boundary Analysis

Overall accuracy does not show whether ABED improves the boundary that motivates it. A method can raise Unknown precision by predicting Unknown less often, or raise Unknown recall by predicting it more often, and either move can leave the underlying confusion unchanged. We therefore evaluate the Unknown class directly. Figure 5 reports Unknown precision, recall, and F1 under GPT-4.1. ABED reaches the highest Unknown F1 on all three datasets, improving over SymbCoT

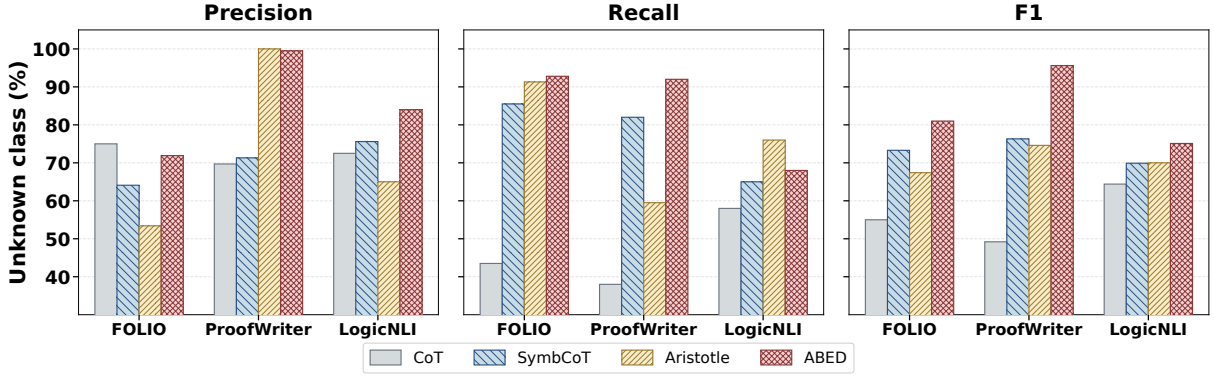


Figure 5: Unknown-class precision, recall, and F1 under GPT-4.1. ABED reaches the highest Unknown F1 on all three datasets. The baselines lean toward either precision or recall and the direction of that lean changes across datasets, whereas ABED stays high on both.

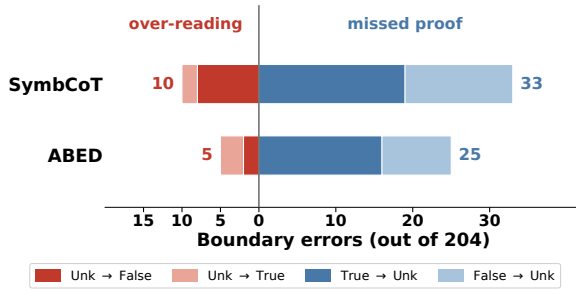


Figure 6: Unknown-boundary errors on FOLIO under GPT-4.1. Bars to the left of zero are gold *Unknown* instances read as determinate, and bars to the right are gold determinate instances left as *Unknown*. ABED shortens both sides.

from 73.3 to 81.0 on FOLIO, from 76.3 to 95.6 on ProofWriter, and from 69.9 to 75.1 on LogicNLI. The two component metrics show that this gain does not come from one side alone. Aristotle reaches 91.3 recall on FOLIO at 53.4 precision, and 100.0 precision on ProofWriter at 59.5 recall, so which axis it favors changes with the dataset. ABED stays high on both axes across all three. The margin tracks how much of the boundary the later stages recover. It is largest on ProofWriter, where Unknown precision reaches 99.5 and the symbolic audit resolves the largest share of unresolved (*Adm*, *Adm*) pairs, and smallest on LogicNLI, where those stages recover less.

Figure 6 separates the two error types behind these numbers on FOLIO, where 69 of the 204 instances are gold Unknown. SymbCoT reads 10 of them as determinate and 8 of these as False, which is the mapping error illustrated in Figure 1. ABED reduces these to 5 and 2. In the opposite direction, gold True or False instances that are left as

Unknown fall from 33 to 25. Both sides shorten, so the improvement is not a shift in how often Unknown is predicted but a change in which instances receive it. The second direction is where the symbolic audit acts. It fires only on instances already diagnosed as open on both sides, and can only promote them when one side contradicts, so it removes false Unknown predictions without introducing new ones.

5 Conclusion

We presented ABED, a training-free framework that reframes open-world logical entailment as missing assumption diagnosis. Instead of reading a label from a single proof-search outcome, ABED asks what assumption is missing for a conclusion to hold, classifies the resulting gap as *Null*, *Contra*, or *Admissible*, and applies this diagnosis symmetrically to a conclusion and its negation, resolving the dual gap pair through verification, consistency alignment, and a restricted symbolic audit. Across four benchmarks and three backbones, ABED consistently improves over prompting-based, symbolic, solver-based, and deductive baselines and remains stable as proof chains grow longer, showing that diagnosing this gap is a more reliable decision signal than committing directly to a proof verdict. A class-wise analysis locates these gains at the boundary the framework targets. ABED obtains the highest Unknown F1 on every three-way dataset, reducing both the cases where an undecided conclusion is read as determinate and the cases where a provable one is left undecided. The gain is thus a change in which conclusions are left open, not in how often the model abstains.

6 Limitations

ABED has several limitations. First, it relies on a fixed inference-time pipeline of LLM calls that includes negation generation, assumption gap estimation, boolean verification, and conditional repair and audit. This increases inference cost relative to single-pass prompting, and the number of calls grows further for cases that trigger consistency repair or the symbolic audit, a cost we did not optimize. Second, the gap diagnosis is produced by an LLM, so an incorrectly estimated bridge can still mislead the verification stage when its errors are not caught by the boolean checks. Third, the symbolic audit is intentionally restricted. It extracts only ground facts, simple rules, disjunctions, and subsumption relations, and it applies a small set of inference rules, so it does not recover proofs that fall outside this fragment. Fourth, our evaluation covers four English logical reasoning benchmarks whose premises are explicitly given and assumed to be internally consistent, and we do not evaluate inconsistent premise sets, multilingual inputs, or settings where premises must first be retrieved. Finally, our main evaluation uses GPT-4.1, GPT-4.1-mini, and DeepSeek-V3.2, all relatively large or API-based models. We report additional results with an open-weight 9B model in the appendix, where the same gains hold, but behavior at smaller scales remains to be examined.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) under Grant No. RS-2023-00273751 funded by the Korean government.

References

- Hirohiko Abe, Risako Ando, Takanobu Morishita, Kentaro Ozeki, Koji Mineshima, and Mitsuhiro Okada. 2024. Abductive reasoning with syllogistic forms in large language models. In *International Conference on Human and Artificial Rationalities*, pages 3–17. Springer.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.
- Errett Bishop. 1967. Foundations of constructive analysis.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Benjamin Callewaert, Simon Vandeveld, and Joost Vennekens. 2025. Verus-lm: a versatile framework for combining llms with symbolic reasoning. *arXiv preprint arXiv:2501.14540*.
- Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert Van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. *arXiv preprint arXiv:2502.15652*.
- Marc Denecker and Antonis Kakas. 2002. Abduction in logic programming. In *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski Part I*, pages 402–436. Springer.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhen-ting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2024. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031.
- Jinwei He and Feng Lu. 2024. Causejudger: Identifying the cause with llms for abductive logical reasoning. *arXiv preprint arXiv:2409.05559*.
- Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 1266–1279.
- Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. Lambada: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6547–6568.
- Jinu Lee and Wonseok Hwang. 2025. Symba: Symbolic backward chaining for structured natural language reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2468–2484.

- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Hanmeng Liu, Zhizhang Fu, Mengru Ding, Ruoxi Ning, Chaoli Zhang, Xiaozhang Liu, and Yue Zhang. 2025a. Logical reasoning in large language models: A survey. *arXiv preprint arXiv:2502.09100*.
- Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Yuting Zeng, Jiaying Wang, Xingyu Wang, Hailong Yang, and Jing Li. 2025b. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10168–10185.
- Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824.
- Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. 2022. Bidirectional language models are also few-shot learners. *arXiv preprint arXiv:2209.14500*.
- Hyun Ryu, Gyeongman Kim, Hyemin S Lee, and Eunho Yang. 2025. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *International Conference on Learning Representations*, volume 2025, pages 24935–24964.
- Moein Salimi, Shaygan Adim, Danial Parnian, Nima Alighardashi, Mahdi Jafari Siavoshani, and Mohammad Hossein Rohban. 2026. Wiring the ‘why’: A unified taxonomy and survey of abductive reasoning in llms. *arXiv preprint arXiv:2604.08016*.
- Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.
- Peiyang Song, Pengrui Han, and Noah Goodman. 2025. A survey on large language model reasoning failures. In *2nd AI for Math Workshop @ ICML 2025*.
- Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9828–9862.
- Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634.
- Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through logicnli. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 2025. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3075.
- Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Wenting Zhao, Justin Chiu, Claire Cardie, and Alexander M Rush. 2023. Abductive commonsense reasoning exploiting mutually exclusive explanations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14883–14896.

A Dataset Details

We evaluate ABED on four logical reasoning benchmarks. For FOLIO, ProofWriter, and ProntoQA, we use the dataset versions released by Logic-LM (Pan et al., 2023), so that our results are directly comparable to prior prompting-based and solver-based work that adopts the same splits. **FOLIO** (Han et al., 2024) is an expert-written natural language reasoning dataset grounded in first-order logic. Each instance pairs a set of premises with a conclusion, and the task is three-way classification over *True*, *False*, and *Unknown*. **ProofWriter** (Tafjord et al., 2021) consists of rule-based reasoning problems over synthetic facts and rules. We use the open-world assumption (OWA) split, in which a conclusion can be *True*, *False*, or *Unknown*. Instances are annotated with the maximum reasoning depth required, which we use in the depth-wise analysis (Section 4.3). **ProntoQA** (Saparov and He, 2022) is a synthetic benchmark of multi-step deductive chains. Each instance is a binary yes/no entailment question, and we evaluate it as a binary task. **LogicNLI** (Tian et al., 2021) is an NLI-style benchmark for first-order logical reasoning, with four labels: entailment, contradiction, neutral, and self-contradiction. We use 300 instances, excluding self-contradiction since it marks internally inconsistent premises outside our open-world setting (Section 3.1). The remaining labels map to *True*, *False*, and *Unknown*.

B Baseline Details

We compare ABED against seven baselines spanning prompting-based, logic-oriented, and symbolic or solver-based approaches. For every baseline, we use the official code released by the original authors, and re-run all methods on our benchmarks under the same backbone models (GPT-4.1 and GPT-4.1-mini) for a fair comparison. We keep each method’s original prompts and hyperparameters as provided in its public implementation, changing only the backbone model and the evaluation data.

Prompting-based. **Vanilla** prompts the backbone model to directly output a label for the conclusion without intermediate reasoning steps. **Zero-shot CoT** (Wei et al., 2022) prepends a chain-of-thought instruction so that the model produces intermediate reasoning before its final answer. These two baselines measure how far a backbone model

Stage	General	ProntoQA
Negation generation	1	1
AGE	2	2
BGV	2	2
CCDA	0–1	0–1
SFC	0–1	0
Total	5–7	5–6

Table 5: Per-stage LLM call budget of ABED. The general setting applies to three-way open-world benchmarks, while ProntoQA uses the same gap diagnosis but collapses non-Null states into the binary no label.

can go without any logic-specific scaffolding.

Logic-oriented. **DetermLR** (Sun et al., 2024) reformulates logical reasoning as a progression from indeterminate to determinate premises, iteratively accumulating and prioritizing derived conditions. **Aristotle** (Xu et al., 2025) introduces a logic-complete decompose-search-resolve framework, with an entailment classifier based on resolution and proof by contradiction.

Symbolic and solver-based. **SymbCoT** (Xu et al., 2024) translates a problem into a symbolic logic representation and performs stepwise reasoning over it before producing a verdict. **Logic-LM** (Pan et al., 2023) converts natural-language premises into a formal representation and delegates inference to an external symbolic solver. **LINC** (Olausson et al., 2023) generates multiple symbolic translations of a problem, runs an external first-order-logic prover on each, and aggregates the results by majority vote. For all baselines, we follow the same evaluation protocol as ABED. FOLIO, ProofWriter, and LogicNLI are evaluated as three-way classification, and ProntoQA as a binary task. When a baseline’s official implementation does not natively support the three-way *True/False/Unknown* setting of a given benchmark, we adapt only its output-parsing layer to the three-way label space, leaving its reasoning procedure unchanged.

C Implementation Details

Models and APIs. We access GPT-4.1 and GPT-4.1-mini through the OpenAI API, and DeepSeek-V3.2 through the Novita AI API. For all models and all methods, we use greedy decoding with temperature 0.0, a maximum generation length of 2048 tokens, and a fixed random seed of 42, so that results are as reproducible as the underlying APIs

Method	FOLIO	PW	Pronto	LogicNLI
Direct	291	298	257	439
CoT	297	304	263	445
Logic-LM	2,011	1,463	1,630	2,472
LINC	2,110	2,369	2,359	3,054
SymbCoT	9,886	11,831	10,204	10,307
ABED	13,155	12,087	7,224	12,987
DetermLR	19,192	8,830	38,325	22,276
Aristotle	42,602	54,652	17,377	44,386

Table 6: Average total token usage per example under GPT-4.1 across four benchmarks. Total tokens include both input prompt tokens and output completion tokens.

Method	FOLIO	PW	Pronto	LogicNLI
Direct / CoT	1.0	1.0	1.0	1.0
Logic-LM / LINC	1.0	1.0	1.0	1.0
SymbCoT	3.0	3.0	3.0	3.0
ABED	5.8	5.8	5.0	5.0
DetermLR	13.1	8.8	45.9	16.6
Aristotle	12.8	16.3	9.0	14.7

Table 7: Average number of LLM calls per example under GPT-4.1.

allow.

Number of LLM calls. ABED is an inference-time pipeline, so its cost is determined by the number of LLM calls per instance rather than by training. The number of calls is not fixed. It depends on how far an instance progresses through the pipeline, since later stages are triggered only for unresolved cases. Table 5 summarizes the per-stage call budget. Negation generation, AGE, and BGV are always executed, contributing five calls per instance: one for negation generation, two for AGE (one per target), and two for BGV (one per target). CCDA adds at most one call, and only when the post-BGV dual pair is invalid and matches a repair target. SFC adds at most one call for symbolic parsing of the premises, and only when both directions still remain ADM after the preceding stages. In total, ABED uses between five and seven LLM calls per instance, with the higher end reached only by the relatively few cases that require consistency repair or symbolic auditing. For ProntoQA, the upper bound is six calls since SFC is not invoked under the binary evaluation setting.

D Cost Analysis

Total token usage. Since ABED is an inference-time pipeline, we analyze its computational cost in terms of token usage. Table 6 reports the average total tokens per example across the four benchmarks,

Method	FOLIO	PW	Pronto	LogicNLI
Direct	0.9	1.0	0.5	0.8
CoT	0.6	0.7	0.6	0.8
Logic-LM	2.6	3.2	3.2	5.2
LINC	1.2	1.0	1.0	1.3
SymbCoT	21.6	12.7	20.1	18.2
ABED	27.1	20.8	23.8	24.9
DetermLR	19.6	11.1	48.2	22.7
Aristotle	35.2	39.8	27.5	69.7

Table 8: Average wall-clock time per example under GPT-4.1, reported in seconds. Wall-clock time includes the end-to-end inference time for each method.

Method	FOLIO	PW	Pronto	LogicNLI	Total
Direct / CoT	0.06	0.18	0.13	0.13	0.50
Logic-LM	0.41	0.88	0.82	0.74	2.84
LINC	0.43	1.42	1.18	0.92	3.95
SymbCoT	2.02	7.10	5.10	3.09	17.31
ABED	2.68	7.25	3.61	3.90	17.44
DetermLR	3.92	5.30	19.16	6.68	35.06
Aristotle	8.69	32.79	8.69	13.32	63.49

Table 9: Total token usage on the full development sets under GPT-4.1, reported in millions of tokens. The dataset sizes are 204 for FOLIO, 600 for ProofWriter, 500 for ProntoQA, and 300 for LogicNLI.

including both input prompt tokens and output completion tokens. ABED operates in a token regime comparable to SymbCoT, the strongest symbolic baseline. Averaged over the four datasets, ABED uses about 11.4K tokens per example, while SymbCoT uses about 10.6K, a difference of under 10%, despite ABED achieving higher accuracy on every benchmark (Table 1). At the same time, ABED is substantially cheaper than more decomposition-heavy logic-oriented baselines. DetermLR uses about 22.2K tokens per example on average, nearly twice the cost of ABED, and Aristotle uses about 39.8K tokens on average, more than three times the cost of ABED. The gap is especially large on FOLIO, ProofWriter, and LogicNLI, where Aristotle exceeds 40K tokens per example. Prompting baselines such as Direct and CoT are far cheaper, using only a few hundred tokens, but trail ABED in accuracy by a wide margin. Overall, ABED occupies a favorable point on the cost–accuracy trade-off: it remains close to the token budget of the strongest comparable symbolic baseline while improving accuracy, and avoids the much heavier cost of decomposition-based reasoning methods.

LLM calls Table 7 reports the average number of LLM calls per example under GPT-4.1. Single-pass methods such as Direct, CoT, Logic-LM, and LINC require only one LLM call, while Symb-

CoT uses three calls for translation, planning, and solving. ABED requires 5.0 to 5.8 calls per example because it performs dual-target assumption diagnosis, verification, and conditional repair. Although this is higher than single-pass prompting, it remains substantially lower and more stable than decomposition-heavy methods such as DetermLR and Aristotle. For example, DetermLR requires 45.9 calls per example on ProntoQA and Aristotle requires 16.3 calls on ProofWriter, whereas ABED stays below six calls across all datasets.

Wall-clock time. Table 8 reports the average wall-clock time per example under GPT-4.1. As expected, single-pass methods are the fastest because they require only one LLM call. ABED is slower than Direct, CoT, Logic-LM, and LINC due to its multi-stage inference-time pipeline, but its runtime remains comparable to other structured reasoning methods. In particular, ABED takes 20.8 seconds on ProofWriter and 24.9 seconds on LogicNLI, while Aristotle requires 39.8 and 69.7 seconds on the same datasets. Compared with DetermLR, ABED is slower on FOLIO and ProofWriter, but faster on ProntoQA and comparable on LogicNLI. Compared with Aristotle, ABED is faster on ProofWriter and LogicNLI, slightly faster on ProntoQA, and slower on FOLIO. These results show that ABED introduces additional latency over single-pass prompting, but avoids the substantially higher runtime of some decomposition-heavy reasoning pipelines.

Full development token usage Table 9 reports the total token usage on the full development sets under GPT-4.1. Across all four datasets, ABED uses 17.44M tokens, which is nearly identical to SymbCoT’s 17.31M tokens. In contrast, DetermLR uses 35.06M tokens, about twice the total cost of ABED, and Aristotle uses 63.49M tokens, more than three times the cost of ABED. This shows that ABED remains in the same overall token regime as the strongest symbolic baseline, while avoiding the much larger total token budget of decomposition-heavy logic-oriented methods.

E Self-consistency.

Table 10 reports the effect of applying self-consistency to direct prediction. SC provides small gains on FOLIO, ProofWriter, and ProntoQA, but the improvements are limited and it even slightly degrades performance on LogicNLI. This suggests

that simply sampling or repeating the same reasoning process does not resolve the core failure mode in open-world logical reasoning. When the model repeatedly maps an underdetermined case to a decisive label, majority voting tends to reinforce the same error rather than diagnose the missing entailment relation.

F Statistical Analysis

Table 14 reports pairwise statistical significance tests under GPT-4.1. We evaluate the statistical significance of performance differences between ABED and each baseline using paired bootstrap resampling. For each dataset and baseline, we repeatedly resample the evaluation set with replacement for 10,000 iterations and compute the accuracy difference between ABED and the corresponding baseline. The reported difference corresponds to the mean improvement over the bootstrap samples, and the 95% confidence interval is obtained from the 2.5th and 97.5th percentiles of the bootstrap distribution.

ABED shows statistically significant improvements over every baseline on all four datasets, with consistently positive 95% confidence intervals. The strongest gains appear on ProofWriter, while ProntoQA shows smaller but still significant improvements due to near-saturated baseline performance. These results indicate that ABED’s accuracy gains are statistically reliable rather than incidental.

G Example of a Full ABED Trace

To illustrate how the four stages of ABED interact on a concrete instance, we walk through a complete trace on FOLIO_dev_4, shown in Figure 7. The gold label is *Unknown*, and ABED predicts *Unknown*.

Setup. The premises describe generic rules about employees, managers, lunch locations, and remote work (P1–P6), together with a disjunctive fact about James (P7): James is either a manager who appears in the company today, or not a manager and absent today. The conclusion is *C*: *James has lunch in the company*. Crucially, P7 leaves the disjunction unresolved, so the truth of *C* depends on which case holds, and the premises alone do not fix this.

Stage 1: Generation of $\neg C$. ABED first generates the strict negation $\neg C$: *James does not have*

Dataset	CoT	SC	Δ
FOLIO	67.7	72.5	+4.8
PW	60.5	63.7	+3.2
Pronto	92.0	94.2	+2.2
LogicNLI	68.3	67.3	-1.0

Table 10: Effect of self-consistency. SC improves over CoT prediction on most datasets, but the gains remain limited and do not resolve systematic reasoning errors.

lunch in the company. The two targets C and $\neg C$ are then diagnosed independently.

Stage 2: Assumption Gap Diagnosis. For the C track, AGE reasons backward and proposes the bridging hypothesis *James is a manager and appears in the company today*, the case of P7 under which C can be settled: from being a manager, P6 gives that James does not work remotely from home; by contraposition of P4, James does not have lunch at home; and with P3, James must have lunch in the company. For the $\neg C$ track, AGE proposes the opposite case, *James is not a manager and does not appear in the company today*, under which P1 and P2, by contraposition, give that James does not have lunch in the company. BGV then checks each proposal with the four boolean tests. On both tracks, the target is neither derivable from P (b_1) nor contradicted by it (b_2), and the bridge is not already supported by P (b_3) but does close the gap when added (b_4). By the BGV priority rule, both tracks are therefore assigned *Adm*.

Stage 3: CCDA. The verified dual pair is (Adm, Adm) , which is already a valid pair. CCDA performs no repair, since neither an incomplete asymmetric pair nor a logically invalid symmetric conflict is present.

Stage 4: SFC. Because the pair remains (Adm, Adm) , ABED invokes the symbolic audit. SFC extracts the rule set from the premises and runs forward chaining, but with no grounding fact that fixes which case of P7 holds for James, neither C nor $\neg C$ yields a one-sided contradiction. The pair is therefore kept as (Adm, Adm) .

Outcome. Mapping the final pair (Adm, Adm) to a label gives *Unknown*, matching the gold label. The trace shows the intended behavior of ABED: rather than committing to a label from a single proof attempt, it diagnoses that each side of the disjunction is genuinely open, and reports the underdetermination explicitly.

H Example of an SFC Audit

Figure 8 shows a case where the symbolic audit recovers a proof that the language-level diagnosis misses. The conclusion C (*the bald eagle is round*) is in fact derivable, but only through a five-step forward chain. Because AGE and BGV reason over natural language and do not trace this chain to its end, both directions are left *Adm*; mapping the pair directly to *Unknown* would therefore be a false negative. The repaired pair remains (Adm, Adm) , which triggers SFC.

SFC first extracts a restricted symbolic form $\mathcal{S}(P) = (\mathcal{F}, \mathcal{R}, \mathcal{D}, \mathcal{L})$, retaining only ground facts, simple implication rules, ground disjunctions, and subsumption relations. For this instance the relevant content is a handful of ground facts (e.g., $COLD(squirrel)$, $KIND(squirrel)$, $COLD(bald_eagle)$) and four implication rules; there are no disjunctions or subsumptions to extract. A deterministic forward chainer then saturates the premises under each assumption. Assuming $\neg C$, repeated modus ponens fires the chain $COLD(sq) \rightarrow CHASESCAT(sq) \rightarrow ROUND(sq) \rightarrow EATSBE(sq) \rightarrow ROUND(bald_eagle)$, which derives $ROUND(bald_eagle)$ and directly contradicts the assumption $\neg ROUND(bald_eagle)$.

This is a one-sided contradiction $OC(\neg C)$: only the $\neg C$ assumption collapses, so the audit promotes the pair from (Adm, Adm) to $(Null, Contra)$ and ABED predicts *True*. The example illustrates the intended role of SFC—a conservative check that converts a small number of false *Unknown* predictions into correct determinate ones, without acting as a general-purpose solver: it fires only on unresolved (Adm, Adm) pairs and changes the decision only when one side, and not the other, saturates to a contradiction.

I Prompts

ABED is a training-free pipeline, and all stages are realized through constrained LLM prompts. We list the five prompts here so that the pipeline can be reproduced. Each prompt fixes the task, gives explicit logical rules, and constrains the output to a fixed schema; placeholders such as {premises} and {target} are filled in at run time.

Negation (Stage 1). Figure 9 shows the prompt that generates the proposition-level negation $\neg C$ of the input conclusion. It specifies rewrite rules for quantified and conditional statements and requires a single-sentence output with entity names

ABED			
Gold \ Pred	True	False	Unknown
True	51	5	16
False	5	49	9
Unknown	3	2	64

SymbCoT			
Gold \ Pred	True	False	Unknown
True	49	4	19
False	0	49	14
Unknown	2	8	59

Table 11: Gold-by-prediction confusion matrices on FOLIO under GPT-4.1.

Dataset	# Inst.	CoT	ABED	Δ
FOLIO	204	63.7	77.5	+13.8
ProofWriter	600	54.8	89.0	+34.2

Table 12: Accuracy with Qwen3.5-9B as the backbone, under the same protocol as the main experiments.

preserved.

Assumption Gap Estimation (Stage 2). Figure 10 shows the AGE prompt. It diagnoses the gap between the premises and a target as NULL, CONTRADICTION, or ADMISSIBLE, and for admissible cases proposes a compact bridge hypothesis h .

Boolean Gap Verification (Stage 2). Figure 11 shows the BGV prompt. It audits the bridge h proposed by AGE through the four boolean checks b_1 – b_4 , without generating or revising h .

Consistency-Constrained Dual Alignment (Stage 3). Figure 12 shows the CCDA conflict-recheck prompt. It is invoked only for symmetric determinate conflicts and directly re-evaluates whether P entails C and $\neg C$.

Symbolic Parser (Stage 4). Figure 13 shows the parser prompt used by SFC. It converts the premises and conclusion into a normalized symbolic form; a separate deterministic forward chainer then operates on this output.

J FOLIO Confusion Matrices

Table 11 reports gold-by-prediction confusion matrices on FOLIO under GPT-4.1 for ABED and SymbCoT, the strongest baseline on this dataset. These are the counts summarized in Figure 6. Of the 204 instances, 72 are gold True, 63 are gold False, and 69 are gold Unknown.

Bridge type	n	%
Atomic fact gap or other	154	75.5
Disjunction / case selection	20	9.8
Multi-step / multi-premise	18	8.8
Explicit single-rule antecedent	12	5.9

Table 13: Manual annotation of the bridges diagnosed for all 204 FOLIO instances under GPT-4.1.

K Smaller Open-Weight Backbone

Our main experiments use GPT-4.1, GPT-4.1-mini, and DeepSeek-V3.2, all of which are large or API-based. To check whether the gains hold on a smaller open-weight model, we additionally evaluate Qwen3.5-9B on FOLIO and ProofWriter, the two open-world benchmarks with the largest proportion of gold Unknown instances. We follow the same splits, prompts, evaluation protocol, and greedy decoding setting as in the main experiments, changing only the backbone.

Table 12 reports the results. ABED improves over CoT by 13.8 points on FOLIO and by 34.2 points on ProofWriter. The larger gain on ProofWriter is consistent with Section 4.2, where the improvement is most pronounced when the backbone is weaker and more likely to lose track of multi-step proof chains. Unknown recall on FOLIO also rises from 53.6 to 72.5, so the boundary behavior described in Section 4.6 is preserved at this scale.

L Types of Diagnosed Gaps

AGE produces a bridging hypothesis for each target, and the four boolean checks then decide whether the diagnosed gap is genuine. To characterize what kinds of gaps this step actually surfaces, we manually annotated the bridges diagnosed for all 204 FOLIO instances in our main GPT-4.1 run.

Table 13 reports the distribution. Most bridges are atomic fact gaps, where a single missing proposition separates the premises from the target. The remaining categories are structurally different. In 8.8% of cases the bridge is a composite hypothesis that requires several premises before the target follows, and in 9.8% it is a choice of which branch of a stated disjunction holds. The FOLIO trace in Appendix G is an instance of the latter, where each bridge is one case of the disjunction in P7 and the premises leave that case undecided. Only 5.9% of bridges are the antecedent of a single rule whose consequent is the target.

The annotation is descriptive and covers one dataset, so it does not measure how the gap type relates to accuracy. It indicates that the diagnosed gaps are not of a single kind, and that a substantial fraction involve premise composition or case selection rather than a single missing antecedent.

Dataset	Baseline	ABED Acc.	Diff.	95% CI	p-value
FOLIO	Vanilla	80.4	+10.3	[6.4, 14.7]	< 0.001***
	CoT		+12.7	[8.3, 17.6]	< 0.001***
	Logic-LM		+4.4	[2.0, 7.4]	< 0.001***
	SymbCoT		+3.4	[1.0, 5.9]	0.001**
	LINC		+8.8	[4.9, 12.7]	< 0.001***
	Aristotle		+10.3	[6.4, 14.7]	< 0.001***
	DetermLR		+9.8	[5.9, 14.2]	< 0.001***
ProofWriter	Vanilla	95.5	+34.7	[30.8, 38.5]	< 0.001***
	CoT		+35.0	[31.2, 38.8]	< 0.001***
	Logic-LM		+22.2	[18.2, 26.2]	< 0.001***
	SymbCoT		+16.7	[13.7, 19.7]	< 0.001***
	LINC		+5.3	[2.5, 8.3]	< 0.001***
	Aristotle		+9.0	[6.8, 11.3]	< 0.001***
	DetermLR		+16.5	[13.7, 19.5]	< 0.001***
ProntoQA	Vanilla	100.0	+24.0	[20.4, 27.8]	< 0.001***
	CoT		+8.0	[5.8, 10.4]	< 0.001***
	Logic-LM		+3.6	[2.0, 5.4]	< 0.001***
	SymbCoT		+1.8	[0.8, 3.0]	< 0.001***
	LINC		+2.0	[0.8, 3.4]	< 0.001***
	Aristotle		+0.8	[0.2, 1.6]	0.039*
	DetermLR		+1.0	[0.2, 2.0]	0.009**
LogicNLI	Vanilla	74.7	+6.7	[4.0, 9.7]	< 0.001***
	CoT		+6.3	[3.7, 9.3]	< 0.001***
	Logic-LM		+9.0	[6.0, 12.3]	< 0.001***
	SymbCoT		+4.3	[2.3, 6.7]	< 0.001***
	LINC		+7.3	[4.7, 10.3]	< 0.001***
	Aristotle		+7.3	[4.7, 10.3]	< 0.001***
	DetermLR		+13.0	[9.3, 16.7]	< 0.001***

Table 14: Pairwise statistical significance of ABED over each baseline. ABED accuracy is shown for each dataset, and **Diff.** denotes the percentage-point improvement over the corresponding baseline. Confidence intervals are 95% intervals for the accuracy difference.

Premises P :

- P1: If an employee schedules a meeting with customers, then they appear in the company today.
P2: If an employee has lunch in the company, then they schedule a meeting with customers.
P3: Every employee has lunch in the company or has lunch at home.
P4: If an employee has lunch at home, then they work remotely from home.
P5: If an employee is in other countries, then they work remotely from home.
P6: If an employee is a manager, then they do not work remotely from home.
P7: James is a manager and appears in the company today, or James is not a manager and does not appear in the company today.

Conclusion C : *James has lunch in the company.* **Generated $\neg C$:** *James does not have lunch in the company.*

P7 is a disjunction over two cases for James, and the premises do not fix which case holds.

C track (target: *James has lunch in the company*)

AGE bridge h	<i>James is a manager and appears in the company today.</i>
(closes via)	P6, P4, P3 give “has lunch in company”
b_1 : target derivable from P ?	No
b_2 : target contradicts P ?	No
b_3 : bridge already supported by P ?	No
b_4 : $P \cup \{h\}$ derives target?	Yes
Verified gap	<i>Adm</i>

$\neg C$ track (target: *James does not have lunch in the company*)

AGE bridge h	<i>James is not a manager and does not appear in the company today.</i>
(closes via)	P1, P2 give “no lunch in company”
b_1 : target derivable from P ?	No
b_2 : target contradicts P ?	No
b_3 : bridge already supported by P ?	No
b_4 : $P \cup \{h\}$ derives target?	Yes
Verified gap	<i>Adm</i>

Each bridge is one case of the P7 disjunction; since the premises leave the case undecided, both gaps stay open.

CCDA: valid pair, no repair. **SFC:** no one-sided contradiction, so the pair is kept. $(Adm, Adm) \rightarrow$ **Unknown** (gold: *Unknown*).

Figure 7: A successful ABED trace on FOLIO (FOLIO_dev_4). The premises contain a disjunction (P7) with two cases for James, and the conclusion’s truth depends on which case holds. For each target, AGE recovers the case that would close it: *James is a manager and appears today* closes C (via P6, P4, and P3, James must have lunch in the company), and *James is not a manager and does not appear* closes $\neg C$ (via P1 and P2, James cannot have lunch in the company). The boolean checks b_1 – b_4 confirm each bridge is consistent with the premises but not entailed by them. Since the premises do not decide the disjunction, both directions are *Adm*, CCDA keeps the pair, and SFC finds no one-sided contradiction, so ABED correctly predicts *Unknown*.

Conclusion C : *The bald eagle is round.* **Generated $\neg C$:** *The bald eagle is not round.* AGE + BGV: (*Adm*, *Adm*)
Reaching $\text{ROUND}(\text{bald_eagle})$ needs a five-step chain that the language-level diagnosis does not complete, so both directions are left *Adm* and SFC is triggered.

Step 1 — Symbolic extraction $\mathcal{S}(P) = (\mathcal{F}, \mathcal{R}, \mathcal{D}, \mathcal{L})$:

$\mathcal{F}$ *ground facts*: $\text{COLD}(\text{squirrel})$, $\text{KIND}(\text{squirrel})$, $\text{COLD}(\text{bald_eagle})$, ...

$\mathcal{R}$ *implication rules*: $\text{COLD}(x) \rightarrow \text{CHASESCAT}(x)$; $\text{CHASESCAT}(x) \wedge \text{KIND}(x) \rightarrow \text{ROUND}(x)$; $\text{ROUND}(x) \rightarrow \text{EATSBE}(x)$; $\text{EATSBE}(x) \wedge \text{COLD}(be) \rightarrow \text{ROUND}(be)$

$\mathcal{D}$ *ground disjunctions*: none

$\mathcal{L}$ *subsumptions*: none

Step 2 — Forward chaining under $P \cup \{\neg C\}$ (modus ponens):

$$\underbrace{\text{COLD}(sq)}_{\mathcal{F}} \xrightarrow{\text{COLD} \rightarrow \text{CHASESCAT}} \text{CHASESCAT}(sq) \xrightarrow{\wedge \text{KIND}(sq)} \text{ROUND}(sq) \\ \xrightarrow{\text{ROUND} \rightarrow \text{EATSBE}} \text{EATSBE}(sq) \xrightarrow{\wedge \text{COLD}(be)} \text{ROUND}(be)$$

Step 3 — Contradiction. The chain derives $\text{ROUND}(be)$, but $\neg C$ asserts $\neg \text{ROUND}(be)$. Witness: $\{\text{ROUND}(be), \neg \text{ROUND}(be)\} \Rightarrow \perp$. The C side stays consistent.

$$\text{OC}(\neg C): (\text{Adm}, \text{Adm}) \xrightarrow{\text{SFC}} (\text{Null}, \text{Contra}) \rightarrow \mathbf{True} \text{ (gold: True)}$$

Figure 8: An SFC audit recovering a missed proof on ProofWriter ($be = \text{bald_eagle}$, $sq = \text{squirrel}$). AGE and BGV leave both directions *Adm* because the language-level diagnosis does not trace the five-step chain to ROUND . SFC extracts a restricted symbolic form $\mathcal{S}(P) = (\mathcal{F}, \mathcal{R}, \mathcal{D}, \mathcal{L})$ —ground facts, implication rules, ground disjunctions, and subsumption relations—and forward-chains under each assumption. Under $P \cup \{\neg C\}$ the chain derives $\text{ROUND}(be)$, contradicting the assumption, while the C side stays consistent. This one-sided contradiction promotes the pair to (*Null*, *Contra*), so ABED predicts *True* rather than a false *Unknown*.

You produce the proposition-level negation of a statement.

Rules:

- Negate at the proposition level (not just adding “not” or “It is not the case that”).
- Preserve quantifiers correctly:
 - “All X are Y” $\rightarrow$ “Some X is not Y.”
 - “Some X is Y” $\rightarrow$ “No X is Y.”
 - “If A then B” $\rightarrow$ “A holds but B does not.”
- For a simple atomic statement “S P”, produce “S does not P.” with grammatical agreement.
- Keep entity names (proper nouns) unchanged.
- Output exactly one English sentence ending with a period.

STATEMENT:

“{target}”

Figure 9: The constrained prompt used in Stage 1 to generate the proposition-level negation $\neg C$ of the input conclusion. The placeholder {target} is replaced with the conclusion C .

You are a strict logical diagnostician under open-world reasoning.

Your task is to diagnose the gap between the PREMISES and the TARGET. Do not produce a final task label. Do not rely on closed-world reasoning.

Choose exactly one gap_type:

1. NULL
The TARGET is already entailed by the PREMISES.
Use this when the premises logically force the TARGET, even if the wording is different.
2. CONTRADICTORY
The TARGET is incompatible with the PREMISES.
Use this only when assuming the TARGET leads to a contradiction with the premises.
3. ADMISSIBLE
The TARGET is compatible with the PREMISES but not entailed by them.
In this case, propose a compact bridge hypothesis h that would make the TARGET follow if h were added to the premises.
The bridge h must not already be entailed by the premises.

Important rules:

- Do not infer the converse of a conditional.
- Do not use closed-world reasoning: absence of proof is not false.
- Use contraposition only for clearly stated strict logical rules.
- Do not introduce outside commonsense or unstated background facts.
- If the proposed bridge is already entailed by the premises, the gap is NULL, not ADMISSIBLE.
- ADMISSIBLE means the target is left open; it is not evidence that the target is true.

Reasoning steps:

1. Check whether the TARGET is entailed by the PREMISES. If yes, output NULL.
2. Check whether the TARGET contradicts the PREMISES. If yes, output CONTRADICTORY.
3. Otherwise, identify the weakest compact bridge h that would make the TARGET follow.
If h is not already entailed and is consistent with the premises, output ADMISSIBLE.

PREMISES:
{premises}

TARGET:
“{target}”

Output strictly according to the provided JSON schema.

Figure 10: The Assumption Gap Estimation (AGE) prompt used in Stage 2. AGE diagnoses the gap between the premises and the target as NULL, CONTRADICTORY, or ADMISSIBLE, and proposes a bridge hypothesis h for admissible cases. The placeholders {premises} and {target} are filled in at run time.

You are a meticulous logical auditor for open-world natural-language FOL problems.

Your task is to verify a proposed bridge hypothesis h for a TARGET. Do not generate a new bridge hypothesis. Do not revise, replace, or strengthen h . Only audit the given h .

Definitions:

- P: the numbered premises below. Treat them as the only source of facts and rules.
- TARGET: the hypothesis being diagnosed in this direction.
- h : the proposed bridge hypothesis from AGE. If h is empty, vague, or unusable, do not infer a replacement. Mark bridge-related checks as false and explain briefly.

Answer exactly four boolean checks:

- 1) `premises_entails_target`
Does P alone entail TARGET?
- 2) `target_contradicts_premises`
Is TARGET inconsistent with P? Equivalently, does P union {TARGET} lead to a contradiction?
- 3) `bridge_entailed_by_premises`
Is h already entailed by P? If yes, h is a pseudo-gap rather than an independent bridge.
- 4) `premises_plus_bridge_entails_target`
Does P union { h } entail TARGET? This should be true only if h is usable, consistent with P, and sufficient for TARGET.

Rules:

- Use only the premises and their strict logical consequences.
- Do not use closed-world reasoning: absence of proof is not falsity.
- Do not infer unstated converses from conditionals.
- Use contraposition only for clearly stated strict logical rules.
- Do not introduce outside commonsense or unstated background facts.
- Lexical overlap is not entailment.
- Trivial reformulation is allowed only when it preserves the same stated meaning.
- If h is already entailed by P, it may help close the target, but it is not an independent gap.

PREMISES:
{premises}

TARGET:
“{target}”

PROPOSED_BRIDGE_H:
{hypothesis}

Output strictly according to the provided JSON schema.

Figure 11: The Boolean Gap Verification (BGV) prompt used in Stage 2. BGV audits the bridge hypothesis h proposed by AGE through the four boolean checks b_1 – b_4 , without generating or revising h . The placeholders {premises}, {target}, and {hypothesis} are filled in at run time.

You are a consistency-recheck auditor for dual abductive entailment.

This recheck is used only for symmetric determinate conflicts: (NULL, NULL) or (CONTRADICTION, CONTRADICTION).

Assume the premise set P is internally consistent. Under open-world FOL, the valid final dual states are:

1. (NULL, CONTRADICTION) $\Rightarrow P$ entails C $\Rightarrow$ True
2. (CONTRADICTION, NULL) $\Rightarrow P$ entails not- C $\Rightarrow$ False
3. (ADMISSIBLE, ADMISSIBLE) $\Rightarrow P$ entails neither C nor not- $C \Rightarrow$ Unknown

The prior diagnosis produced a symmetric conflict. Your task is not to repair the previous trace. Your task is to directly recheck whether P entails C and whether P entails not- C .

PRIOR DIAGNOSIS:

```
gap_C = {gap_c}   for C = "{target_c}"
gap_not_C = {gap_nc}   for not-C = "{target_nc}"
```

Procedure:

1. Decide $p_entails_c$: Does P alone entail C ?
2. Decide $p_entails_not_c$: Does P alone entail not- C ?

Use strict open-world reasoning:

- Use only the premises and their strict logical consequences.
- Do not use closed-world reasoning: absence of proof is not falsity.
- Do not infer unstated converses.
- Use contraposition only for clearly stated strict logical rules.
- Do not introduce outside commonsense or unstated background facts.
- Trivial reformulation is allowed only when it preserves the same stated meaning.

Decision mapping:

- If $p_entails_c$ is true and $p_entails_not_c$ is false:
 $truth_class = \text{"True"}$, $repaired_pair = [\text{"NULL"}, \text{"CONTRADICTION"}]$
- If $p_entails_c$ is false and $p_entails_not_c$ is true:
 $truth_class = \text{"False"}$, $repaired_pair = [\text{"CONTRADICTION"}, \text{"NULL"}]$
- If both are false:
 $truth_class = \text{"Unknown"}$, $repaired_pair = [\text{"ADMISSIBLE"}, \text{"ADMISSIBLE"}]$
- If both appear true: recheck carefully, since this contradicts the assumption that P is consistent. If the conflict cannot be resolved from P alone, set both to false:
 $truth_class = \text{"Unknown"}$, $repaired_pair = [\text{"ADMISSIBLE"}, \text{"ADMISSIBLE"}]$

PREMISES:

```
{premises}
```

PRIOR TRACE for C (sanity-check only; do not trust it over P):

```
{trace_c_excerpt}
```

PRIOR TRACE for not- C (sanity-check only; do not trust it over P):

```
{trace_nc_excerpt}
```

Output strictly according to the provided JSON schema.

Figure 12: The CCDA conflict-recheck prompt used in Stage 3. It is invoked only for symmetric determinate conflicts, (NULL,NULL) or (CONTRADICTION,CONTRADICTION), which are impossible under a consistent premise set. The recheck directly re-evaluates whether P entails C and whether P entails $\neg C$, and maps the result to a valid dual pair. The placeholders are filled in at run time.

You are a SYMBOLIC PARSER. You convert natural-language logic premises and a conclusion into a normalized symbolic form. You DO NOT solve the problem. A separate deterministic forward chainer will run on your output.

OUTPUT RULES (mandatory):

1. Use SHORT, snake_case predicate names. Strip articles, demonstratives, and adjectives that do not carry logical content (unless the predicate IS the adjective). Example: "John has a child" -> has_child(john).
2. Use the SAME predicate spelling everywhere.
3. For named entities, use lowercased snake_case constants. For variables in universal premises, use single lowercase letters x, y, z.
4. For each premise, decide its TYPE:
 - GROUND FACT -> append to premise_facts.
 - UNIVERSAL RULE -> append to premise_rules with variable in args.
 - DISJUNCTION -> append to premise_disjunctions (>= 2 options).
 - DEFINITIONAL -> append to subsumptions.
5. For "If A or B then C", split into TWO rules (A -> C) and (B -> C).
6. For "X is EITHER A OR B", emit ONE entry in premise_disjunctions with options [A(x), B(x)]. Do NOT emit a rule with BOTH consequents - that asserts "A AND B", which is logically WRONG.
7. For "A iff B", emit BOTH rules (A -> B) and (B -> A).
8. For "no X are Y", emit a rule $X(x) \rightarrow \text{not } Y(x)$.
9. For "every / all / each / any X is Y", emit $X(x) \rightarrow Y(x)$.
10. For ANY existential statement, emit GROUND FACTS with fresh skolem constants. An existential is NEVER a universal disjunction; it is always one or more ground skolem facts.
11. CONCLUSION: emit conclusion_atom_c for C and conclusion_atom_nc for its negation, same pred and args, neg flipped.

PREDICATE NORMALIZATION TIPS:

- "is X" -> is_X ; "has X" -> has_X
- Prefer neg=true on a positive predicate over inventing "not_X".
- The predicate dictionary MUST list every predicate used, with a gloss.

PREMISES (numbered):

{premises}

CONCLUSION (C):

{conclusion}

Return JSON only, matching the schema exactly. Do NOT include reasoning prose; only the JSON object.

Figure 13: The symbolic parser prompt used in Stage 4 (SFC). The LLM only parses the premises and conclusion into a normalized symbolic form; a separate deterministic forward chainer operates on the output. The placeholders {premises} and {conclusion} are filled in at run time.

Algorithm 1 ABED Pipeline for a Single Logical Entailment Instance

Input: premises $P = \{p_1, \dots, p_n\}$, conclusion C

Output: final answer $\hat{A} \in \{True, False, Unknown\}$

(1) Complementary Target Construction

$\neg C \leftarrow \text{Negate}(C)$

▷ proposition-level negation

$\mathcal{H} \leftarrow \{C, \neg C\}$

▷ bidirectional targets

(2) Assumption Gap Diagnosis

For each $H \in \mathcal{H}$:

$(\gamma_0(P, H), h) \leftarrow \text{AGE}(P, H)$

▷ initial gap and bridge

$b_1 \leftarrow [P \vdash H]$

▷ target is entailed

$b_2 \leftarrow [P \cup \{H\} \vdash \perp]$

▷ target is contradictory

$b_3 \leftarrow [P \vdash h]$

▷ bridge is supported

$b_4 \leftarrow [P \cup \{h\} \vdash H]$

▷ bridge closes the gap

$$\hat{\gamma}(P, H) \leftarrow \begin{cases} \text{Null}, & b_1, \\ \text{Contra}, & b_2, \\ \text{Null}, & b_3 \wedge b_4, \\ \text{Adm}, & \text{otherwise.} \end{cases}$$

▷ BGV priority rule

(3) Dual-Pair Consistency Alignment

$\hat{\Gamma}(P, C) \leftarrow (\hat{\gamma}(P, C), \hat{\gamma}(P, \neg C))$

▷ verified dual gap pair

$\Gamma_r(P, C) \leftarrow \text{CCDA}(\hat{\Gamma}(P, C))$

▷ consistency alignment and repair

(4) Local Symbolic Audit

If $\Gamma_r(P, C) = (\text{Adm}, \text{Adm})$:

$\mathcal{S}(P) \leftarrow (\mathcal{F}, \mathcal{R}, \mathcal{D}, \mathcal{L})$

▷ restricted symbolic form

$$\Gamma^*(P, C) \leftarrow \begin{cases} (\text{Null}, \text{Contra}), & \text{OC}(\neg C), \\ (\text{Contra}, \text{Null}), & \text{OC}(C), \\ (\text{Adm}, \text{Adm}), & \text{otherwise.} \end{cases}$$

▷ SFC audit

Otherwise: $\Gamma^*(P, C) \leftarrow \Gamma_r(P, C)$

(5) Final Label Mapping

$$\hat{A} \leftarrow \begin{cases} \text{True}, & \Gamma^*(P, C) = (\text{Null}, \text{Contra}), \\ \text{False}, & \Gamma^*(P, C) = (\text{Contra}, \text{Null}), \\ \text{Unknown}, & \Gamma^*(P, C) = (\text{Adm}, \text{Adm}). \end{cases}$$

▷ final prediction

return $\hat{A}$
