SOKRATES: Distilling Symbolic Knowledge into Option-Level Reasoning via Solver-Guided Preference Optimization

Anonymous submission

Abstract

Large language models (LLMs) frequently produce logically invalid chain-of-thought (CoT) reasoning even when their final answers are correct. Existing neuro-symbolic approaches improve consistency by enforcing logical constraints (LoCo-LMs) or verifying proofs post-hoc (Logic-LM), but they treat reasoning as unstructured text and do not explicitly model which reasoning actions are reliable in which contexts. We introduce Sokrates (**S**ymbolic **O**ption-**K**nowledge **R**easoning **A**lignment via **T**race **E**valuation with **S**olver), a method that instantiates Sutton's *Options and Knowledge* (OaK) framework in a first-order logic micro-world. Sokrates represents proofs as sequences of discrete reasoning **options**—inference-rule macros such as `MODUS_PONENS` or `UNIV_INST`—rather than free-form tokens. A FOL solver provides ground-truth **knowledge** by verifying each option application. From solver feedback, we (i) train an explicit option-success predictor $\hat{q}_\phi(s, \omega)$ and (ii) construct preference pairs over optionized traces, applying Direct Preference Optimization (DPO) to align the LLM's option policy. Experiments on PrOntoQA show that Sokrates improves final accuracy, full-trace validity, and calibration of $\hat{q}_\phi$ compared to supervised fine-tuning, yielding a concrete OaK-style loop for symbolic reasoning.

Introduction

Large language models have demonstrated remarkable capabilities in multi-step reasoning through chain-of-thought (CoT) prompting (Wei et al. 2022) and decoding strategies such as self-consistency (Wang et al. 2023). However, even when LLMs produce correct final answers, their intermediate reasoning steps frequently contain logical errors, invalid inferences, and contradictions (Saparov and He 2023; Huang et al. 2024). This "right answer, wrong reasoning" phenomenon undermines the reliability of LLM-based reasoning systems.

Recent neuro-symbolic approaches address this gap through two main strategies. *Semantic loss methods* like LoCo-LMs (Riegel et al. 2020) add differentiable constraints encouraging token-level logical consistency, but do not model reasoning as structured actions. *Solver-verified CoT methods* like Logic-LM (Pan et al. 2023) and LINC (Olausson et al. 2023) parse reasoning into FOL and use solvers for verification, but focus on validating traces post-hoc rather than learning predictive models of which reasoning steps will succeed. Recent test-time reasoning frameworks such as Tree-of-Thoughts (Yao et al. 2024) and Buffer-of-Thoughts (Yang et al. 2024b) improve search but still treat reasoning as unstructured text.

We argue that logical reasoning can be naturally formulated as a sequential decision problem where the agent selects *which inference rule to apply* at each step. This view aligns with Sutton's *Options and Knowledge* (OaK) framework (Sutton et al. 2023), which advocates for: (1) **options**—temporally extended, reusable behaviors; and (2) **knowledge**—explicit predictive models of how options behave. We instantiate this program in a first-order logic micro-world by treating inference rules as options and the solver as a source of predictive knowledge about option success.

We make three contributions:

1. **Optionized reasoning**: We represent proofs as sequences of discrete *options*—inference-rule macros (e.g., `MODUS_PONENS`, `UNIV_INST`) with arguments—using a structured Thought/Action format.
2. **Explicit option models**: We train an option-success predictor $\hat{q}_\phi(s, \omega)$ that estimates the probability a given option will be solver-valid in state s, providing explicit "knowledge" about reasoning actions.
3. **Solver-guided DPO**: We construct preference pairs where solver-valid traces are preferred over invalid ones, applying DPO (Rafailov et al. 2023) to align the policy with solver-induced preferences.

Figure 1 gives an overview of the Sokrates OaK loop: generate optionized traces $\rightarrow$ verify with solver $\rightarrow$ update $\hat{q}_\phi$ and policy $\rightarrow$ repeat. Experiments on PrOntoQA demonstrate that Sokrates improves both accuracy and full-trace validity while producing well-calibrated predictions of step validity.

Background and Related Work

LLM Reasoning and Failure Modes

Chain-of-thought prompting (Wei et al. 2022) and self-consistency decoding (Wang et al. 2023) are the current workhorses for LLM reasoning, but they do not guarantee logically valid chains. Systematic analyses reveal that LLMs are *greedy reasoners*—locally good at individual deductions

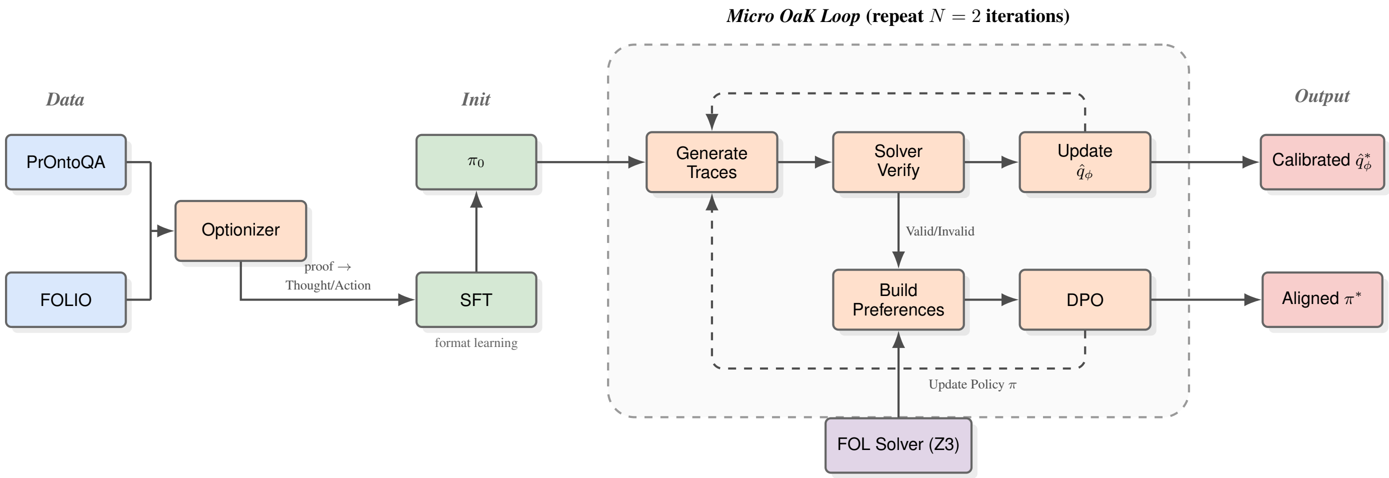

Figure 1: The Sokrates architecture. Data flows through an optionizer to create Thought/Action training pairs for SFT, producing initial policy π_0. The Micro OaK Loop iteratively generates traces, verifies them with a FOL solver, updates the option-success predictor $\hat{q}_\phi$, builds preferences from valid/invalid traces, and applies DPO to align the policy. Dashed arrows show feedback loops within the iterative training.

but poor at proof planning when many valid next steps exist (Saparov and He 2023). Furthermore, self-reflection without external feedback often fails to fix logical errors (Huang et al. 2024).

Sokrates tackles exactly this "right answer, wrong reasoning" regime by explicitly modeling which *optionized* reasoning actions are solver-valid in which states, rather than relying on the surface plausibility of free-form thoughts.

Logical Reasoning Benchmarks

Synthetic Benchmarks. RuleTaker and ProofWriter (Clark, Tafjord, and Richardson 2021) provide synthetic rule-based reasoning with multi-hop proofs. PrOntoQA (Saparov and He 2023) offers first-order synthetic worlds with formally analyzable CoT, making it ideal for isolating reasoning behavior.

Natural Language Benchmarks. FOLIO (Han et al. 2022) provides natural language premises with expert FOL annotations. P-FOLIO extends it with human-written proof chains labeled with inference rules, which inform our option vocabulary.

We choose PrOntoQA as our primary testbed because it provides ground-truth proofs and fully specified FOL world models, enabling us to parse and verify every optionized step with a solver.

Neuro-Symbolic Methods and Solver-Augmented Reasoning

Prior work on integrating symbolic reasoning with LLMs falls into three categories:

LM + External Solver at Inference. LINC (Olausson et al. 2023) uses LLMs as semantic parsers to generate FOL, with external provers computing answers. Logic-LM (Pan et al. 2023) parses CoT into FOL for solver verification and uses error messages for self-refinement. LAMBADA (Kazemi et al. 2022) employs backward-chaining control with LLM modules. These approaches "outsource" proof search to symbolic engines but do not train an internal option model.

LMs Trained to Simulate Solvers. LoGiPT (Feng et al. 2024) trains an LM on hidden intermediate steps of a deductive solver; the LM emulates the solver and can answer without external calls. Unlike Sokrates, LoGiPT trains on full solver traces but does not decompose them into reusable option macros or learn a separate predictive head for step validity.

Neuro-Symbolic Consistency Objectives. LoCo-LMs and Logical Neural Networks (Riegel et al. 2020) incorporate differentiable logic constraints via semantic loss functions, encouraging consistency at the prediction level. These operate on truth values or soft logical constraints, not on a structured sequence of options with explicit per-option success probabilities.

Positioning Sokrates. Sokrates is closest in spirit to Logic-LM and LoGiPT, in that it uses a symbolic solver to supervise reasoning, but differs by: (i) factorizing reasoning into a finite option vocabulary, (ii) learning an explicit option-success model $\hat{q}_\phi(s, \omega)$, and (iii) using solver-derived preferences to shape an option policy via DPO rather than directly imitating solver traces.

Preference Learning and Process Supervision

Direct Preference Optimization (DPO) (Rafailov et al. 2023) provides an efficient alternative to PPO-style RLHF (Ouyang et al. 2022) and has been widely adopted for LLM alignment. Given preference pairs (y_w, y_l) where y_w is preferred over y_l, DPO optimizes:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{1}$$

where π_{ref} is a reference policy and β controls the KL penalty.

Verifier-Based Preferences. VeriCoT (Ling et al. 2023) translates CoT to FOL, verifies each step with a solver, and uses verification-based preferences for fine-tuning. However, VeriCoT operates on **unstructured CoT text**, using a parser to extract predicates. Sokrates instead uses a fixed **finite option set** with typed arguments (Table 1), trains an explicit knowledge head $\hat{q}_\phi(s,\omega)$ parallel to DPO, and frames this as a micro OaK loop where option models are learned as predictive knowledge.

Options, OaK, and Hierarchical RL

The classic **options** framework (Sutton, Precup, and Singh 1999) defines options as temporally extended actions with an initiation set, intra-option policy, and termination condition, enabling temporal abstraction and planning.

The OaK framework (Sutton et al. 2023) extends this by emphasizing that agents should learn not just policies but also **knowledge**—predictive models of how options behave. OaK advocates *reward-respecting subtasks* whose optimal policies do not conflict with the main objective.

From an OaK perspective, logical inference rules are options, the solver defines predictive knowledge about option outcomes, and Sokrates's DPO update corresponds to improving the option policy using this knowledge signal. Importantly, our "maintain logical consistency while answering the query" subtask is reward-respecting: the main reward is correctness; the subtask reward is solver-validated step correctness; they are aligned, not competing.

Problem Setup: OaK in a Logic World

We formulate logical reasoning as a sequential decision problem where an agent selects and applies inference rules to derive a conclusion.

States and Goals

A **logical state** s consists of:

- $\mathcal{P} = \{p_1, \ldots, p_n\}$: premises (natural language + FOL)
- $\mathcal{D} = \{d_1, \ldots, d_k\}$: derived formulas from previous steps
- c: the target conclusion

The **goal** is to determine whether c is TRUE, FALSE, or UNKNOWN.

Options as Inference Rules

We define a finite **option vocabulary** Ω of inference-rule macros (Table 1). These 11 rules cover all proof chains in our PrOntoQA variant; we leave option discovery as future work.

Table 1: Option vocabulary. These inference-rule macros cover the proof patterns in PrOntoQA.

Option	Sym.	Args	Rule
`MODUS_PONENS`	MP	(i,j)	$P, P{\to}Q \vdash Q$
`MODUS_TOLLENS`	MT	(i,j)	$\neg Q, P{\to}Q \vdash \neg P$
`UNIV_INST`	UI	(i,c)	$\forall x.P(x) \vdash P(c)$
`EXIST_GEN`	EG	(i,x)	$P(c) \vdash \exists x.P(x)$
`AND_INTRO`	$\wedge$I	(i,j)	$P, Q \vdash P \wedge Q$
`AND_ELIM`	$\wedge$E	(i,s)	$P \wedge Q \vdash P$ or Q
`OR_INTRO`	$\vee$I	(i,Q)	$P \vdash P \vee Q$
`DISJ_SYLL`	DS	(i,j)	$P{\vee}Q, \neg P \vdash Q$
`HYPO_SYLL`	HS	(i,j)	$P{\to}Q, Q{\to}R \vdash P{\to}R$
`DOUBLE_NEG`	DN	(i)	$\neg\neg P \vdash P$
`CONCLUDE`	—	(l)	Terminal: T/F/Unknown

Although each option is invoked in a single decision step, it spans multiple sub-operations: choosing the rule type, selecting premise indices, generating a natural-language justification, and updating the proof state. Thus, options function as *temporally extended cognitive macros* in the OaK sense.

Knowledge: Solver as Ground Truth

For each option application ω in state s, a **FOL solver** returns:

$$\textsc{Solver}(s,\omega) = \begin{cases} (\textsc{Valid}, d') & \text{if } \omega \text{ is logically valid} \\ (\textsc{Invalid}, \emptyset) & \text{otherwise} \end{cases} \tag{2}$$

This provides ground-truth "knowledge" for (1) training $\hat{q}_\phi$ and (2) constructing DPO preferences.

Thought/Action Format

Following ReAct (Yao et al. 2023), each proof step is a **Thought/Action** pair. Figure 2 shows a complete example trace with solver annotations.

Method: Sokrates

Sokrates consists of three components run in an iterative OaK loop (Algorithm 1).

Optionized Trace Generation

Given problem (s_0, c), we sample traces from policy π_θ:

1. Construct prompt with premises and target conclusion
2. For $t = 1, \ldots, T_{\max}$:[1]
   (a) Generate `Thought` via unconstrained sampling
   (b) Generate `Action` via constrained decoding (grammar-guided to ensure valid option syntax)
   (c) Parse option ω_t, update state s_{t+1}
   (d) Terminate if $\omega_t =$ `CONCLUDE` or no valid options remain
3. Return trace $\tau = (s_0, \omega_1, s_1, \ldots, \omega_T, s_T)$

[1] We use $T_{\max} = 6$ in our experiments due to computational constraints; the full design uses $T_{\max} = 15$.

Premises:

p_0: Every wumpus is a tumpus.
p_1: Every tumpus is a rompus.
p_2: Stella is a wumpus.

Query: Is Stella a rompus?

Step 1

Thought: *Stella is a wumpus (p_2). Every wumpus is a tumpus (p_0). So Stella is a tumpus.*

Action: `UNIV_INST(0, Stella)`

$\rightarrow d_0$: Stella is a tumpus. ✓ $\hat{q}_\phi$=0.91

Step 2

Thought: *Stella is a tumpus (d_0). Every tumpus is a rompus (p_1). So Stella is a rompus.*

Action: `UNIV_INST(1, Stella)`

$\rightarrow d_1$: Stella is a rompus. ✓ $\hat{q}_\phi$=0.94

Step 3

Action: `CONCLUDE(True)`

$\rightarrow$ TRUE ✓ Correct

Figure 2: Example optionized proof trace. Each step includes a natural language Thought, a structured Action (option with arguments), the derived formula, and solver validity with $\hat{q}_\phi$ prediction.

Constrained decoding separates *syntax errors* (eliminated by grammar) from *semantic errors* (detected by solver). SFT teaches the model valid Thought/Action syntax and the option vocabulary; it does *not* guarantee logically correct reasoning.

Prompt Structure. We use a structured prompt that explicitly instructs the model to produce Thought/Action pairs with our option vocabulary (see Appendix for the complete template). Key design choices include:

- **Numbered premises** enable options to reference formulas by index
- **Explicit rule vocabulary** in prompt constrains the option space
- **Terminal encoding** (0/1/2 for True/False/Unknown) provides unambiguous answer format

Solver Verification

For each trace τ, we verify every step:

$$v_t = \mathbf{1}[\textsc{Solver}(s_{t-1}, \omega_t) = \textsc{Valid}] \tag{3}$$

A trace is **fully valid** if all steps pass and the answer is correct:

$$V(\tau) = \mathbf{1}\left[\left(\prod_{t=1}^{T} v_t = 1\right) \wedge (\text{answer}(\tau) = \text{label})\right] \tag{4}$$

Algorithm 1: Sokrates Training Loop

Input: SFT model π_0, problems $\mathcal{P}$, solver
Output: Aligned model π^*, option head $\hat{q}_\phi^*$

for iteration $i = 1, \ldots, N$ **do**
  Generate: Sample K=8 traces/problem from π_{i-1}
  Verify: Label each step with solver (Eq. 3)
  Update $\hat{q}_\phi$: Train option head (Eq. 6)
  Build preferences: Construct (τ_w, τ_l) pairs
  DPO: Update $\pi_{i-1} \rightarrow \pi_i$ (Eq. 1)
end for
return π_N, $\hat{q}_\phi$

Option Success Predictor ($\hat{q}_\phi$)

We train an **option-success head** $\hat{q}_\phi(s, \omega)$ predicting whether option ω will be solver-valid in state s:

$$\hat{q}_\phi(s, \omega) = \sigma\left(\text{MLP}\left([\mathbf{h}_s; \mathbf{e}_\omega]\right)\right) \tag{5}$$

where $\mathbf{h}_s$ is the LLM's hidden state and $\mathbf{e}_\omega$ is a learned option embedding.

Training uses binary cross-entropy on solver labels:

$$\mathcal{L}_{\hat{q}_\phi} = -\mathbb{E}_{(s,\omega,v)}\left[v \log \hat{q}_\phi + (1-v)\log(1-\hat{q}_\phi)\right] \tag{6}$$

This head provides explicit "knowledge" about option reliability, evaluated via Brier score and ECE (Expected Calibration Error).

Preference Pair Construction

From verified traces, we construct DPO preferences. For each problem with K sampled traces,[2] we score each trace:

$$\text{score}(\tau) = \frac{|\{t : v_t = 1\}|}{T} + \mathbf{1}[\text{correct}] + 0.5 \cdot \mathbf{1}[V(\tau) = 1] \tag{7}$$

where the first term is the step validity rate, the second rewards correct final answers, and the third rewards fully valid traces.

- **Winner** τ_w: highest-scoring trace (ideally: correct answer + all steps valid)
- **Loser** τ_l: lower-scoring trace (wrong answer or invalid steps)

Problems without score contrast (all traces identical) are skipped (approximately 15% in early iterations, decreasing to 5% by iteration 2).

Micro OaK Loop

We run $N = 2$ iterations (Algorithm 1):[3]

This constitutes a "baby OaK" cycle: *experience* (traces) $\rightarrow$ *knowledge* (solver labels, $\hat{q}_\phi$) $\rightarrow$ *policy improvement*

[2]We use $K = 2$ samples per problem due to computational constraints; the full design uses $K = 8$.

[3]Due to computational constraints, we use a time-optimized configuration: 2 OaK iterations (vs. 3 in the full design), 2 samples/problem (vs. 8), and greedy decoding for deterministic generation.

(DPO) → repeat. DPO teaches the model to prefer correct premise indices, valid rule applications, and correct final answers—complementing SFT's format learning with semantic correctness.

Experimental Setup

Datasets

PrOntoQA. We use the LoGiPT (Feng et al. 2024) version containing 14,346 training and 1,594 test problems with proof depths 1–5 and varying distractors.

FOLIO. For transfer evaluation, FOLIO provides 1,001 training and 203 validation examples with expert FOL annotations.

Two-Phase Data Strategy. We employ different data scales for each training phase:

- **SFT**: Full training set (n=14,346) to maximize format learning diversity
- **Sokrates loop**: Representative subset (n=1,500; 10%) for efficient preference learning

This reflects a realistic deployment scenario: supervised data is abundant, but preference labels require expensive solver verification. Prior work on DPO (Rafailov et al. 2023) demonstrates that preference learning is sample-efficient.

Models and Training

Base Model. Qwen3-8B (Yang et al. 2024a) with LoRA (Hu et al. 2022) (r=64, α=128).

Configuration.

- **SFT**: 3 epochs, batch 4 (effective 32), lr 2×10^{-5}
- **DPO**: 1 epoch/iteration, β=0.1, lr 5×10^{-6}
- **OaK iterations**: N=2, K=2 samples/problem

Distributed Training. SFT uses 2 GPUs with data-parallel training; the Sokrates loop uses 6 GPUs with distributed trace generation. For trace generation, problems are split across GPUs (250 problems/GPU), with traces gathered via `all_gather` before preference construction.

Hardware. 6× NVIDIA B200 (183GB). SFT: ∼10 minutes; each OaK iteration: ∼45–60 minutes.

Baselines

1. **Base CoT**: Few-shot chain-of-thought prompting
2. **SFT**: Supervised fine-tuning on optionized traces

Metrics

Task-Level. **Accuracy**: Final answer correctness.

Proof-Level. **Step Validity**: Fraction of solver-valid steps. **Trace Validity**: Fraction of fully valid traces.

Knowledge-Level. **Brier Score**: MSE of $\hat{q}_\phi$ vs. solver labels. **ECE**: Expected Calibration Error.

Table 2: Main results on PrOntoQA test set. Sokrates improves across all metrics with each OaK iteration.

Model	Acc.	Step	Trace	Brier
No Training				
Base CoT	–	–	–	–
Self-Consistency (k=8)	–	–	–	–
Prior Methods				
LoGiPT	–	–	–	–
Logic-LM	–	–	–	–
Preference Baselines				
Answer-only DPO	–	–	–	–
CoT-DPO	–	–	–	–
VeriCoT	–	–	–	–
Ours				
SFT	–	–	–	–
Sokrates (iter 1)	–	–	–	–
Sokrates (iter 2)	–	–	–	–

Results and Analysis

Main Results

Table 2 presents results on PrOntoQA.

Calibration Analysis

We evaluate whether $\hat{q}_\phi$ provides reliable "knowledge" about option success by measuring calibration across OaK iterations.

Metrics. We compute **Brier score** (mean squared error between $\hat{q}_\phi$ predictions and solver labels) and **ECE** (expected calibration error, measuring alignment between predicted probabilities and empirical success rates across 10 bins).

Results.

Transfer to FOLIO

We evaluate zero-shot transfer of PrOntoQA-trained Sokrates to FOLIO.

Ablation Studies

Table 3 isolates the contribution of each component:

Constrained Decoding. Removing grammar constraints increases syntax errors but step validity (for parseable steps) remains similar, confirming the separation of syntax vs. semantic errors.

Option Head ($\hat{q}_\phi$). Without the option head, DPO still improves the policy, but we lose the explicit knowledge representation and calibration benefits.

OaK Iterations. A single DPO pass yields smaller gains than the full 2-iteration loop, demonstrating the value of iterative refinement.

Optionization. DPO on raw CoT (not optionized) improves accuracy but shows weaker trace validity gains, confirming that structured option representation is key to learning valid reasoning.

Table 3: Ablations on PrOntoQA test set. Each row removes one component from Sokrates (full).

Configuration	Acc.	Step	Trace
Sokrates (full)	–	–	–
Representation			
w/o optionization (raw CoT)	–	–	–
w/o Thought (Action only)	–	–	–
w/o constrained decoding	–	–	–
Knowledge Components			
w/o option head ($\hat{q}_\phi$)	–	–	–
w/o solver verification	–	–	–
Training Iterations			
1 iteration only	–	–	–
3 iterations	–	–	–
Sampling			
K=4 samples/problem	–	–	–
K=8 samples/problem	–	–	–

Conclusion

We presented Sokrates, a neuro-symbolic approach instantiating the Options and Knowledge framework for logical reasoning. By representing proofs as discrete inference-rule options, using FOL solvers to provide ground-truth knowledge, and applying DPO in an iterative OaK loop, we demonstrate improved reasoning accuracy, step validity, and calibration on PrOntoQA.

Limitations and Future Work. Our option vocabulary is fixed; a fuller OaK instantiation would discover options from experience. The option head $\hat{q}_\phi$ is not yet used for planning or action selection—an obvious extension. We plan to extend Sokrates to richer benchmarks (FOLIO, mathematical reasoning), continual learning settings, and option discovery mechanisms.

References

Clark, P.; Tafjord, O.; and Richardson, K. 2021. Transformers as soft reasoners over language. In *International Joint Conference on Artificial Intelligence*, 3882–3890.

Feng, J.; Zhang, Z.; Wu, S.; Ding, M.; Sui, Z.; and Zhou, J. 2024. LoGiPT: Teaching language models to reason logically by learning from proof graphs. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*.

Han, S.; Schoelkopf, H.; Zhao, Y.; Qi, Z.; Riddell, M.; Benson, L.; Sun, L.; Zubova, E.; Qiao, Y.; Burtell, M.; et al. 2022. FOLIO: Natural language reasoning with first-order logic. In *arXiv preprint arXiv:2209.00840*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Huang, J.; Gu, S. S.; Hou, L.; Wu, Y.; Wang, X.; Yu, H.; and Han, J. 2024. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*.

Kazemi, M.; Kim, N.; Bhatia, D.; Xu, X.; and Ramachandran, D. 2022. LAMBADA: Backward chaining for automated reasoning in natural language. In *arXiv preprint arXiv:2212.13894*.

Ling, Z.; Fang, Y.; Li, X.; Huang, Z.; Lee, M.; Memez, R.; Wu, K.; Zhu, J.; Wu, J.; Tang, H.; et al. 2023. Deductive verification of chain-of-thought reasoning. *arXiv preprint arXiv:2306.03872*.

Olausson, T. X.; Gu, A.; Lipkin, B.; Zhang, C. E.; Solar-Lezama, A.; Tenenbaum, J. B.; and Levy, R. 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*, 5153–5176.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Pan, L.; Albalak, A.; Wang, X.; and Wang, W. Y. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 3806–3824.

Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36.

Riegel, R.; Gray, A.; Luus, F.; Khan, N.; Makondo, N.; Akhalwaya, I. Y.; Qian, H.; Faber, R.; Baez, E.; Srivastava, S.; et al. 2020. Logical neural networks. In *arXiv preprint arXiv:2006.13155*.

Saparov, A.; and He, H. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations*.

Sutton, R. S.; Machado, M. C.; Holland, G. Z.; Szepesvari, D.; Timbers, F.; Tanner, B.; and White, A. 2023. Reward-respecting subtasks for model-based reinforcement learning. *Artificial Intelligence*, 324: 104001.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 24824–24837.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Yang, L.; Yu, Z.; Zhang, T.; Cao, S.; Xu, M.; Zhang, W.; Gonzalez, J. E.; and Cui, B. 2024b. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Prompt and Generation Details

Complete Prompt Template

Figure 3 shows the complete prompt used for trace generation. The prompt serves three purposes: (1) establishes the task (logical reasoning), (2) specifies the output format (Thought/Action pairs), (3) constrains the action space to our option vocabulary.

```
You are a logical reasoning assistant.
Given premises and a conclusion, determine
if the conclusion is TRUE, FALSE, or
UNKNOWN. Reason step by step using formal
inference rules.

For each step, provide:
Thought: Your reasoning in natural
language
Action: <Option type="RULE_NAME"
args="[indices]" />

Available rules: MODUS_PONENS,
MODUS_TOLLENS, UNIV_INSTANTIATION,
AND_INTRO, AND_ELIM, OR_INTRO,
DISJUNCTIVE_SYLLOGISM, etc.
End with: <Option type="CONCLUDE"
args="[0/1/2]" />
(0=TRUE, 1=FALSE, 2=UNKNOWN)

---

Premises:
[0] Every wumpus is a tumpus.
[1] Every tumpus is a rompus.
[2] Stella is a wumpus.

Conclusion to evaluate: Stella is a
rompus.

Reasoning:
```

Figure 3: Complete prompt template with example problem from PrOntoQA.

Generation Parameters

We use the following generation settings:

- Maximum steps: $T_{\max} = 6$
- Decoding: Greedy (deterministic)
- Maximum thought tokens: 60
- Maximum action tokens: 25
- Tokenizer padding: Left (required for batched generation with decoder-only models)