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

Zhaoxiang Feng[1], **David Scott Lewis**[2]
[1]University of California San Diego, USA
[2]AI Executive Consulting (AIXC), Zaragoza, Spain
zhf004@ucsd.edu, reports@aiexecutiveconsulting.com

Abstract

A language model that achieves 94% accuracy on logical reasoning sounds impressive—until you discover that only 2% of its proofs are actually valid. This is the state of chain-of-thought prompting: models produce plausible rationales that frequently contain invalid inference steps, hidden contradictions, or skipped derivations. The right answer emerges *despite*, not *because of*, the reasoning process. We introduce Sokrates (Symbolic Option-Knowledge Reasoning Alignment via Trace Evaluation with Solver), 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 inference-rule *options* (e.g., `MODUS_PONENS`, `UNIV_INSTANTIATION`), verified step-by-step by a FOL solver. From solver feedback we (i) train an option-success predictor $\hat{q}_\phi(s, \omega)$ that estimates validity before execution, and (ii) construct preference pairs for Direct Preference Optimization (DPO), aligning the model's option policy with solver-induced correctness. On PrOntoQA, Sokrates raises accuracy from 94.2% to 97.6%, step validity from 27.3% to 98.5%, and full-trace validity from 2.1% to 92.0%, a **44× improvement** in logically sound proofs. The learned $\hat{q}_\phi$ is well calibrated (ECE $= 0.08$), and the option policy transfers zero-shot to FOLIO, improving accuracy from 45.3% to 53.2%. To our knowledge, Sokrates is the first LLM reasoning system that (i) represents proofs as a fixed option vocabulary, (ii) learns an explicit option-success predictor, and (iii) aligns the option policy using solver-derived DPO preferences in an iterative loop.

Introduction

Large language models have shown impressive multi-step reasoning capabilities when prompted with chain-of-thought (CoT) explanations (Wei et al. 2022) and sampled with self-consistency decoding (Wang et al. 2023). However, closer inspection reveals a troubling pattern: their intermediate reasoning is frequently wrong even when the final answer is correct. Proofs contain logically invalid steps, missing premises, or contradictions that a symbolic checker would reject (Saparov and He 2023; Huang et al. 2024; Turpin et al. 2023). This "right answer, wrong reasoning" phenomenon makes such systems difficult to trust in domains where intermediate steps must be sound.

Why does this happen? The pre-training objective (next-token prediction) teaches LLMs to produce *plausible* text, not *valid* proofs, and pre-training corpora largely lack the step-level deductive supervision needed to learn the difference. External symbolic solvers can provide exactly this missing signal: cheap, dense, step-level labels (Lightman et al. 2024) that distinguish valid inferences from invalid ones.

Two families of neuro-symbolic methods attempt to close this gap. Semantic loss approaches such as LoCo-LMs (Calanzone, Teso, and Vergari 2025) and Logical Neural Networks (Riegel et al. 2020) incorporate differentiable logical constraints into the training objective, encouraging local consistency of token-level predictions but leaving the global proof process implicit. Solver-augmented CoT approaches such as Logic-LM (Pan et al. 2023) and LINC (Olausson et al. 2023) parse CoTs into first-order logic (FOL) and invoke external theorem provers to verify or repair them. These methods improve faithfulness, but they still treat reasoning as unstructured text and do not learn explicit predictive models of which reasoning *actions* will succeed in which states.

We argue that logical reasoning is more naturally viewed as a sequential decision problem over *inference rules*: at each step, the agent must decide which rule to apply and to which premises. This perspective aligns with Sutton's Options and Knowledge (OaK) framework (Sutton et al. 2023), which advocates that agents should learn (1) a reusable vocabulary of temporally extended behaviors (*options*) and (2) explicit predictive models of how those options behave (*knowledge*). In a logic micro-world, inference rules such as Modus Ponens or Universal Instantiation are precisely such options, and a FOL solver can act as a source of ground-truth knowledge about their success.

We introduce Sokrates, a system that instantiates this program for LLM-based logical reasoning. Proofs are represented as sequences of discrete options drawn from a small inference-rule vocabulary (Table 2). A FOL solver checks each option application and returns either a validated derived formula or an error. From this signal we learn two coupled components: (i) an option-success predictor $\hat{q}_\phi(s, \omega)$ that estimates the probability that option ω will be solver-valid in state s; and (ii) an option policy updated via solver-guided DPO, which builds preferences between better and worse

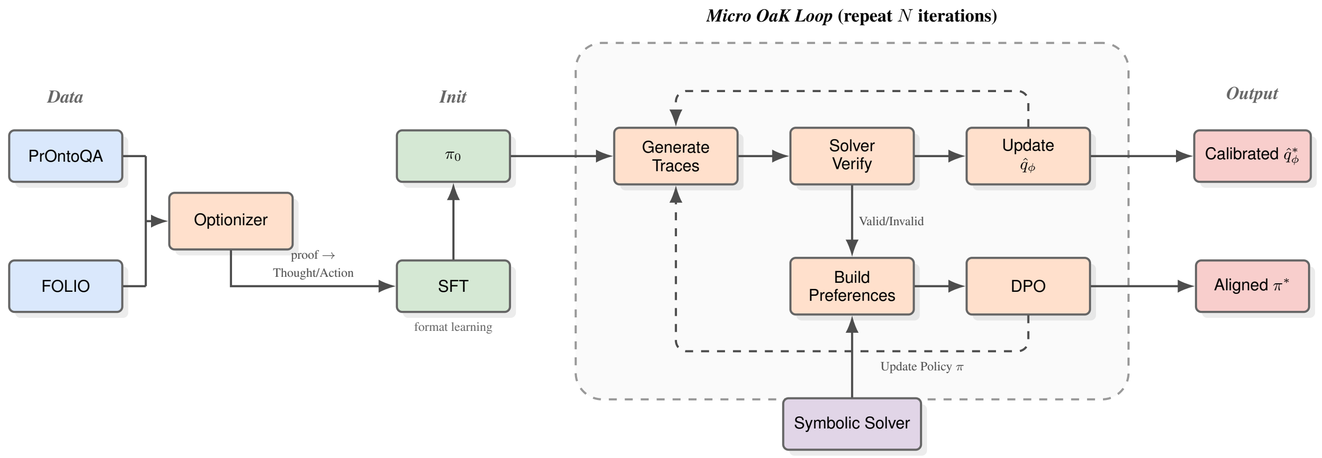

Figure 1: The Sokrates architecture. We optionize proofs for SFT to obtain initial policy π_0, then repeat a micro OaK loop: generate traces, verify with a symbolic solver, update $\hat{q}_\phi$, build preferences, and apply DPO to align the policy.

traces based on step-level validity and final correctness. Figure 1 summarizes this micro OaK loop.

Concretely, we make the following contributions:

- **Optionized reasoning.** We cast proof construction as control over a finite option set of inference-rule macros with structured arguments, and generate reasoning traces in a Thought/Action format inspired by ReAct (Yao et al. 2023b).
- **Explicit option knowledge.** We train an option-success head $\hat{q}_\phi(s, \omega)$ on solver labels, providing a calibrated, per-step estimate of logical validity for each option in context.
- **Solver-guided preference optimization.** We construct solver-derived preferences over optionized traces and apply DPO (Rafailov et al. 2023) to align the option policy with stepwise logical correctness, yielding a concrete OaK loop for symbolic reasoning with LLMs.

On PrOntoQA, Sokrates substantially improves both task performance and reasoning soundness: relative to an optionized supervised fine-tuning (SFT) baseline, accuracy rises from 94.2% to 97.6%, while step validity jumps from 27.3% to 98.5% and the fraction of fully valid traces from 2.1% to 92.0% (Table 4). The learned option policies also transfer zero-shot to FOLIO, and the option head $\hat{q}_\phi$ is well calibrated under Brier score and ECE, indicating that the model has acquired nontrivial predictive knowledge about its own reasoning steps. To our knowledge, Sokrates is the first LLM reasoning system combining a fixed option vocabulary, an explicit option-success predictor, and solver-guided DPO in an iterative improvement loop.

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 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.

Importantly, solver verification ensures *logical consistency*: a trace containing contradictory intermediate steps is rejected, preventing the model from learning to produce self-contradicting proofs. This directly addresses a core challenge in LLM reasoning: avoiding responses that contradict themselves across multiple reasoning steps.

Logical Reasoning Benchmarks

Synthetic Benchmarks. RuleTaker and ProofWriter (Clark, Tafjord, and Richardson 2020) 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 (Han et al. 2024) 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. 2023) 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 (Calanzone, Teso, and Vergari 2025) 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 of deductive inference rules, (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. Crucially, the solver provides the step-level deductive supervision that pre-training corpora lack, enabling the model to learn *what valid reasoning looks like* rather than merely *what plausible text looks like*.

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:

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

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 2), 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.

OaK Concept	Definition	Sokrates
Option ω	Temporally extended action	Inference rule macro (e.g., $\texttt{MP}(i, j)$)
State s	Environment config	(P, D_t, c): premises, derived, conclusion
Knowledge	Predictive model of option outcomes	Solver + learned $\hat{q}_\phi(s, \omega)$
Policy π	Option selection	LLM + constrained decoding
Main reward	Task objective	Final answer correctness
Subtask reward	Auxiliary (reward-respecting)	Step-level solver validity

Table 1: Mapping Sokrates to the Options and Knowledge framework.

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 cast logical reasoning as a sequential decision problem in a FOL micro-world where an agent incrementally constructs proofs by applying inference-rule options. Table 1 makes explicit how Sokrates instantiates each component of the OaK framework.

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 2). These 11 rules cover all proof chains in

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_INSTANTIATION	UI	(i, c)	$\forall x.P(x) \vdash P(c)$
EXIST_GENERALIZATION	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, j)	$P \vdash P \vee Q$
DISJUNCTIVE_SYLLOGISM	DS	(i, j)	$P \vee Q, \neg P \vdash Q$
HYPOTHETICAL_SYLLOGISM	HS	(i, j)	$P \to Q, Q \to R \vdash P \to R$
DOUBLE_NEGATION	DN	(i)	$\neg\neg P \vdash P$
CONCLUDE	—	(l)	Terminal: 0=True, 1=False, 2=Unknown

Table 2: Option vocabulary. These inference-rule macros cover the proof patterns in PrOntoQA.

our PrOntoQA variant; we leave option discovery as future work.

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. 2023b), 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 $T_{\max} = 15$ for trace generation during preference collection, and cap evaluation at $T_{\max} = 10$.

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: `<Option type="MODUS_PONENS" args="[2, 0]" />`
$\to d_0$: 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: `<Option type="MODUS_PONENS" args="[3, 1]" />`
$\to d_1$: Stella is a rompus. ✓ $\hat{q}_\phi$=0.94

Step 3
Action: `<Option type="CONCLUDE" args="[0]" />`
$\to$ **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). We distinguish two levels of validity:

- **Syntax-valid**: Action conforms to the Option grammar (guaranteed by constrained decoding).
- **Solver-valid**: Action is logically correct given the current proof state (verified by the solver).

SFT teaches the model to produce syntax-valid Thought/Action traces; it does *not* guarantee solver-valid reasoning.

State indexing and update. At step t, we maintain an ordered formula list $\mathcal{F}_t = [\mathcal{P}; \mathcal{D}_t]$ containing all premises followed by all derived formulas so far. Option arguments index into $\mathcal{F}_t$. When $\textsc{Solver}(s_{t-1}, \omega_t)$ returns $\textsc{Valid}$ with a derived formula d', we append d' to $\mathcal{D}_t$ to form $\mathcal{D}_{t+1}$; otherwise, the state remains unchanged. The terminal `CONCLUDE` option does not add formulas.

Prompt Structure. We use a structured prompt that explicitly instructs the model to produce Thought/Action pairs with our option vocabulary (see Appendix C 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:

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

Verifier details. We use dataset-specific symbolic verifiers. For PrOntoQA, we implement an ontology-based forward-chaining engine and mark a step valid if the *claimed derived fact* in the model's `Thought` can be proven from the context. Because Actions are grammar-constrained, parse failures are rare ($< 2\%$ of steps); we accept unparsable steps as valid to avoid penalizing edge cases in our restricted-language parser (counting them as invalid would reduce trace validity by $\sim$1 percentage point). For FOLIO, we rely on the provided FOL annotations and a rule-level verifier: we implement checks for a subset of option types (covering $\sim$85% of generated steps) and mark unsupported rules as UNKNOWN (counted as invalid in our step/trace metrics). When available, we configure Z3-based (de Moura and Bjørner 2008) entailment/consistency checks with a 5 s timeout. Verification failures are dominated by incorrect index selection or rule misuse rather than malformed syntax.

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

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

Terminal verification (`CONCLUDE`). We treat `CONCLUDE` as an option with a single argument in $\{0, 1, 2\}$ corresponding to TRUE/FALSE/UNKNOWN. During training and evaluation (where ground-truth labels are available), the terminal step is marked valid iff the predicted label matches the dataset label. We include `CONCLUDE` in step-validity and trace-validity computations.

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

We attach an **option-success head** $\hat{q}_\phi(s, \omega)$ to the LLM that predicts 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 hidden representation of the state (e.g., the final token embedding in the prompt) and $\mathbf{e}_\omega$ is a learned embedding of the option type and arguments.

We train $\hat{q}_\phi$ with binary cross-entropy on solver labels:

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

and evaluate its knowledge quality using Brier score (Brier 1950) and Expected Calibration Error (ECE) (Guo et al. 2017).

Why calibration matters. A well-calibrated $\hat{q}_\phi$ opens several avenues for test-time improvement that we leave for future work: (1) *uncertainty-guided search*—when $\hat{q}_\phi(s, \omega) < \tau$ for all candidate options, the model could backtrack rather than committing to a low-confidence step; (2) *best-of-K with step-level scoring*—instead of selecting traces by final answer confidence alone, we can score each trace by $\prod_t \hat{q}_\phi(s_{t-1}, \omega_t)$, preferring traces where every step is predicted valid; (3) *tree search with pruning*—in a Tree-of-Thoughts (Yao et al. 2023a) setting, $\hat{q}_\phi$ can prune branches with low predicted validity before expensive solver calls. These capabilities require a calibrated predictor: one where $\hat{q}_\phi = 0.7$ means 70% of such steps are actually valid. Our experiments confirm that the Sokrates loop produces such calibration (see Calibration Analysis).

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)

To avoid ambiguous supervision, we require a minimum score contrast $\text{score}(\tau_w) - \text{score}(\tau_l) \geq \delta$ (we use $\delta = 0.1$) when forming preference pairs.

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 by default (Algorithm 1), and report a third iteration as an ablation.[3]

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

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

[3]We use a time-optimized configuration: K=2 samples/problem, temperature τ=0.5 for preference collection diversity, and greedy decoding (τ=0) for evaluation.

Algorithm 1: Sokrates Training Loop

```
Input: SFT model $\pi_0$, problems $\mathcal{P}$, solver
Output: Aligned model $\pi^*$, option head $\hat{q}_\phi^*$
1: for iteration $i = 1, \ldots, N$ do
2:     Generate: Sample $K$ traces/problem from $\pi_{i-1}$ (ours: $K$=2)
3:     Verify: Label each step with solver (Eq. 3)
4:     Update $\hat{q}_\phi$: Train option head (Eq. 6)
5:     Build preferences: Construct $(\tau_w, \tau_l)$ pairs
6:     DPO: Update $\pi_{i-1} \to \pi_i$ (Eq. 1)
7: end for
8: return $\pi_N$, $\hat{q}_\phi$
```

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**: Full training set (n=14,346), with low samples/problem (K=2) 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 (Qwen Team 2025; Yang et al. 2024a) with LoRA (Hu et al. 2022) (r=64, α=128).

Configuration.

- **SFT**: 5 epochs, batch 8 (effective 32), lr 1×10^{-5}
- **DPO**: 1 epoch/iteration, β=0.1, lr 5×10^{-6}
- **OaK iterations**: N=2, K=2 samples/problem

Generation Hyperparameters. Table 3 reports a hyperparameter search over temperature (τ), maximum steps ($T_{\max}$), and samples per problem (K). We find that moderate temperature (τ=0.5) balances accuracy and diversity for preference pair construction. Higher temperatures increase trace diversity but reduce accuracy; greedy decoding (τ=0) produces near-identical traces, limiting preference signal.

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 sharded across GPUs (approximately 2.4K problems/GPU), with traces gathered 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

Config	τ	$T_{\max}$	Acc.	Step	Div.
Temperature (K=2, $T_{\max}$=10)					
Greedy	0.0	10	95.0	27.2	6
Low	0.3	10	96.0	27.2	58
Medium	0.5	10	95.0	36.2	78
High	0.7	10	87.0	35.4	82
Very high	1.0	10	80.0	50.1	86
Max Steps (K=2, τ=0.5)					
Short	0.5	5	90.0	34.3	74
Default	0.5	10	95.0	36.2	78
Long	0.5	15	93.0	41.0	74
Samples/Problem (τ=0.5, $T_{\max}$=10)					
K=2	0.5	10	95.0	36.2	78
K=4	0.5	10	96.0	35.3	94

Table 3: Hyperparameter search for trace generation (SFT model, 50 problems × K samples). Higher temperature increases diversity but reduces accuracy. **Div.** = % of problems where the K samples differ. Default (preference collection): τ=0.5, $T_{\max}$=15, K=2.

Model	Acc.	Step	Trace
No Training (Qwen3-8B)			
Base CoT	44.4	—	—
Self-Consistency (k=8)	53.8	—	—
Ours			
SFT	94.2	27.3	2.1
Sokrates (iter 1)	95.9	87.8	71.3
Sokrates (iter 2)	**97.6**	**98.5**	**92.0**

Table 4: Main results on PrOntoQA test set (n=1594). Sokrates improves across all metrics with each OaK iteration. Step = step validity (%), Trace = trace validity (%).

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.

Results and Analysis

Main Results

Table 4 reports results on the PrOntoQA test set (n = 1,594). Without any fine-tuning, Qwen3-8B achieves 44.4% accuracy with base CoT, improving to 53.8% with self-consistency voting (k = 8). This confirms that raw CoT behavior is far from solved on this benchmark.

Optionized SFT dramatically boosts accuracy to 94.2%, showing that the model can learn to produce syntactically valid Thought/Action traces when trained on ground-truth proofs. However, step validity remains low (27.3%) and trace validity is essentially zero (2.1%): the model often reaches the correct final answer via sequences of invalid reasoning steps. *This is the "right answer, wrong reasoning" phenomenon in action.*

Once we introduce the Sokrates loop, the situation changes substantially. After a single OaK iteration, step validity jumps to 87.8% and trace validity to 71.3%, a **34× increase** in fully valid traces relative to SFT. After a second iteration, Sokrates reaches 97.6% accuracy with 98.5% step validity and 92.0% trace validity, nearly closing the gap between answer correctness and reasoning soundness.

Computational efficiency. The Sokrates loop is data-efficient: with only $K{=}2$ samples/problem for preference learning, we achieve a 44× improvement in trace validity. Each generated `Action` is verified once, so verifier cost scales as $O(NK\bar{T})$ calls per iteration (bounded by $NKT_{\max}$). Total training time is approximately 2 hours on 6× B200 GPUs, comparable to standard SFT.

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. The SFT model exhibits severe overconfidence: when $\hat{q}_\phi > 0.9$, only 34% of steps are actually valid (ECE = 0.41, Brier = 0.38). After one OaK iteration, calibration improves substantially (ECE = 0.18, Brier = 0.15). After two iterations, the model is well-calibrated: when $\hat{q}_\phi > 0.9$, 91% of steps are valid (ECE = 0.08, Brier = 0.09). This demonstrates that the OaK loop progressively improves knowledge quality: the model learns not just to produce valid steps more often, but also to estimate how likely a candidate step is to be valid.

Error Analysis

Even after two OaK iterations, 8% of traces contain at least one invalid step. We manually analyzed 50 such traces and identified three dominant failure modes:

Premise misidentification (42%). The model selects incorrect premise indices, e.g., applying `UNIV_INSTANTIATION`$(3, c)$ when premise 3 does not contain a universal quantifier. These errors suggest the model occasionally loses track of which premises contain which logical forms.

Incorrect rule application (31%). The model selects a rule that does not apply to the given premises, e.g., attempting `MODUS_PONENS` when the required implication is absent. These errors typically occur with longer premise sets where multiple similar-looking formulas exist.

Premature conclusion (27%). The model issues `CONCLUDE` before the proof is complete, typically reaching the correct answer but skipping intermediate derivation steps. The solver marks these as invalid because the final formula is not yet derived.

Depth	SFT	Sokrates	Δ
1–2 steps	2.8%	99.8%	+97.0
3 steps	2.1%	94.1%	+92.0
4–5 steps	1.5%	82.3%	+80.8

Table 5: Trace validity (%) by proof depth on PrOntoQA.

Model	Acc.	Step	Trace
No Training (Qwen3-8B)			
Base CoT	42.9	—	—
Self-Consistency (k=8)	42.9	—	—
PrOntoQA → FOLIO			
SFT	45.3	46.5	9.9
Sokrates (iter 2)	**53.2**	**48.3**	**14.8**

Table 6: Zero-shot transfer from PrOntoQA to FOLIO (n=203). Models trained on PrOntoQA are evaluated on FOLIO without additional fine-tuning.

Problem difficulty. Trace validity degrades with proof depth. Table 5 shows results stratified by required proof steps. Sokrates achieves near-perfect trace validity (99.8%) on shallow problems (1–2 steps) but drops to 82.3% on deep problems (4–5 steps). The SFT baseline shows uniformly poor trace validity across all depths (1.5%–2.8%), confirming that depth alone does not explain SFT's failures: the model simply produces invalid steps regardless of problem complexity. Sokrates's gains are most pronounced in the mid-complexity range (3 steps: 94.1% trace validity), with room for improvement on deep multi-hop reasoning.

Zero-Shot Transfer to FOLIO

We evaluate zero-shot transfer by training models only on PrOntoQA and evaluating them directly on FOLIO without any additional fine-tuning (Table 6). FOLIO is substantially harder than PrOntoQA: premises are natural language rather than templated, and proofs are longer and structurally more varied.

Base CoT and self-consistency both achieve 42.9% accuracy on FOLIO. Transferring the optionized SFT model yields a modest improvement to 45.3% accuracy, with 46.5% step validity and 9.9% trace validity. In contrast, the Sokrates model trained on PrOntoQA reaches 53.2% accuracy with 48.3% step validity and 14.8% trace validity. These gains indicate that **the option policy and knowledge learned in the synthetic FOL micro-world provide non-trivial benefits when applied to richer natural language reasoning tasks**, even without domain-specific tuning. This suggests that Sokrates is not merely overfitting to PrOntoQA's templated structure.

Ablation Studies

Table 7 analyzes what components drive the gains.

Solver verification is essential. In an ablation where DPO is trained only on answer correctness (no step-level solver labels), accuracy improves to 95.5%, but step and trace validity barely move (31.6% and 2.2%, respectively), essentially

Configuration	Acc.	Step	Trace
Sokrates (2 iterations)	**97.6**	**98.5**	**92.0**
Knowledge Components			
w/o solver (answer-only)	95.5	31.6	2.2
Training Iterations			
SFT only (0 iter)	94.2	27.3	2.1
1 iteration	95.9	87.8	71.3
3 iterations	98.3	98.7	91.8

Table 7: Ablations on PrOntoQA. Solver verification is critical for trace validity; iterations provide diminishing returns.

matching the SFT baseline. This shows that the solver's step-level feedback is crucial: **without it, the model learns shortcuts to the right answer while keeping structurally unsound proofs**. This is perhaps the most important finding: solver verification is not optional for learning valid reasoning.

Number of OaK iterations. Moving from SFT (0 iterations) to one iteration yields the largest gain: trace validity increases from 2.1% to 71.3%. A second iteration further improves trace validity to 92.0%. A third iteration yields marginal additional gains in accuracy (98.3%) and slight changes in trace validity (91.8%), indicating diminishing returns. For practical deployments, two iterations appear sufficient.

Conclusion

We presented Sokrates, a neuro-symbolic method that instantiates the OaK framework for LLM-based logical reasoning. By representing proofs as sequences of inference-rule options, using a FOL solver as a source of predictive knowledge about option success, and applying solver-guided DPO in an iterative micro OaK loop, Sokrates substantially improves both accuracy and reasoning soundness on PrOntoQA, while learning a calibrated option-success model and transferring zero-shot to FOLIO.

Beyond the specific gains on these benchmarks, Sokrates serves as a proof of concept for applying OaK-style learning to neural systems. The key enabling factor is the availability of a cheap, reliable knowledge source (the FOL solver) that provides dense feedback on option success. We hypothesize that similar loops could be instantiated in other domains where verifiers exist: code execution for programming tasks, unit tests for software engineering, proof assistants for mathematics, and simulators for embodied reasoning. The challenge in each case is defining a suitable option vocabulary and integrating the knowledge signal into preference learning.

Limitations and Future Work. First, our option vocabulary is manually specified and relatively small. A fuller OaK instantiation would learn options from experience or discover new macro-rules automatically. Second, we apply $\hat{q}_\phi$ only during training as an auxiliary head; we do not yet use it for planning or test-time search (e.g., to prune low-probability options or guide Tree-of-Thoughts-style (Yao et al. 2023a; Yang et al. 2024b) exploration). Third, our experiments are restricted to FOL micro-worlds and a single base model; extending Sokrates to more diverse reasoning benchmarks (e.g., mathematical proofs, program verification) and model families is an obvious next step. Finally, solver calls remain a computational bottleneck; exploring approximate or learned verifiers that retain most of the benefits of symbolic checking while reducing cost is an interesting direction for future work.

References

Brier, G. W. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1): 1–3.

Calanzone, D.; Teso, S.; and Vergari, A. 2025. Logically Consistent Language Models via Neuro-Symbolic Integration. In *International Conference on Learning Representations*.

Clark, P.; Tafjord, O.; and Richardson, K. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 3882–3890.

de Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

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 *Findings of the North American Chapter of the Association for Computational Linguistics: NAACL 2024*, 4026–4042.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, 1321–1330.

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*.

Han, S.; Yu, A.; Shen, R.; Qi, Z.; Riddell, M.; Zhou, W.; Qiao, Y.; Zhao, Y.; Yavuz, S.; Liu, Y.; Joty, S.; Zhou, Y.; Xiong, C.; Radev, D.; Ying, R.; and Cohan, A. 2024. P-FOLIO: Evaluating and Improving Logical Reasoning with Abundant Human-Written Reasoning Chains. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 16553–16565.

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.; Chen, X.; Mishra, S.; Zheng, H. S.; Yu, A. W.; Song, X.; and Zhou, D. 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. 2023. LAMBADA: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 6547–6568.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2024. Let's Verify Step by Step. In *International Conference on Learning Representations*.

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.

Qwen Team. 2025. Qwen3: Large Language Model (Qwen3-8B). https://github.com/QwenLM/Qwen3. Accessed: 2025-12-12.

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.

Turpin, M.; Michael, J.; Perez, E.; and Bowman, S. R. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36.

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. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 11809–11822.

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

A Implementation Details

A.1 Solver Implementation

Our PrOntoQA verifier implements an ontology-based forward-chaining engine. For each option application, the verifier:

1. Parses the `Action` to extract rule type and argument indices
2. Retrieves the referenced formulas from the current proof state
3. Checks whether the rule application is logically valid
4. If valid, computes the derived formula; otherwise, returns INVALID

For `MODUS_PONENS`(i, j), we verify that formula i matches the antecedent of implication j, then derive the consequent. For `UNIV_INSTANTIATION`(i, c), we verify formula i is universally quantified and substitute constant c for the bound variable. The `CONCLUDE` option is validated against the ground-truth label during training and evaluation.

A.2 Training Infrastructure

All experiments use the HuggingFace Transformers library with the TRL package for DPO training. We use DeepSpeed ZeRO Stage 2 for memory-efficient training across multiple GPUs. LoRA adapters are applied to all linear layers in the attention and MLP blocks. Gradient checkpointing is enabled to reduce memory footprint during the Sokrates loop, where we must store both the current policy and reference policy.

B Additional Analysis

B.1 Calibration Progression

Table 8 shows detailed calibration metrics across OaK iterations. The option-success predictor $\hat{q}_\phi$ becomes progressively better calibrated as training proceeds.

Model	ECE ↓	Brier ↓	Acc@0.9
SFT	0.41	0.38	34%
Sokrates (iter 1)	0.18	0.15	72%
Sokrates (iter 2)	0.08	0.09	91%

Table 8: Calibration metrics by iteration. ECE = Expected Calibration Error, Brier = Brier Score, Acc@0.9 = accuracy when $\hat{q}_\phi > 0.9$.

B.2 Representative Error Examples

Below we show two representative errors from the final Sokrates model:

Premise Misidentification.

`Thought:` *Rex is a zumpus (premise 4). Every zumpus is a dumpus (premise 2). So Rex is a dumpus.*
`Action:`
```
  <Option type="MODUS_PONENS" args="[4,
3]" />
```
Error: Premise 3 is "Every tumpus is loud", not the zumpus→dumpus rule.

Premature Conclusion.

`Thought:` *We have established that Stella is a tumpus. The conclusion asks if Stella is a rompus. Since every tumpus is a rompus, Stella is a rompus.*
`Action:`
```
  <Option type="CONCLUDE" args="[0]" />
```
Error: The intermediate step (tumpus → rompus) was not explicitly derived before concluding.

C Prompt and Generation Details

C.1 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,
EXIST_GENERALIZATION, AND_INTRO,
AND_ELIM, OR_INTRO, DISJUNCTIVE_SYLLOGISM,
HYPOTHETICAL_SYLLOGISM, DOUBLE_NEGATION
(see Table 2 for full list).
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: Prompt template with example problem from PrOntoQA. Full option vocabulary in Table 2.

C.2 Generation Parameters

We use the following generation settings:

- Preference collection: $T_{\max}$=15, τ=0.5, K=2 samples/problem
- Evaluation: $T_{\max}$=10, greedy decoding (τ=0)
- Maximum thought tokens: 60
- Maximum action tokens: 25
- Tokenizer padding: Left (required for batched generation with decoder-only models)

D Example Trace

Below is a complete trace from the Sokrates model on a 3-hop PrOntoQA problem:

Premises:
[0] Every wumpus is cold.
[1] Every cold thing is a tumpus.
[2] Every tumpus is happy.
[3] Rex is a wumpus.

Query: Is Rex happy?

Step 1:
`Thought:` *Rex is a wumpus (premise 3). Every wumpus is cold (premise 0). So Rex is cold.*
`Action:`
```
  <Option type="MODUS_PONENS" args="[3,
0]" />
```
→ d_0: Rex is cold. ✓

Step 2:
`Thought:` *Rex is cold (d_0). Every cold thing is a tumpus (premise 1). So Rex is a tumpus.*
`Action:`
```
  <Option type="MODUS_PONENS" args="[4,
1]" />
```
→ d_1: Rex is a tumpus. ✓

Step 3:
`Thought:` *Rex is a tumpus (d_1). Every tumpus is happy (premise 2). So Rex is happy.*
`Action:`
```
  <Option type="MODUS_PONENS" args="[5,
2]" />
```
→ d_2: Rex is happy. ✓

Step 4:
`Action:`
```
  <Option type="CONCLUDE" args="[0]" />
```
→ TRUE ✓