Reasoning Models Lose their Way with Long Deduction

Hadeel Al-Negheimish [1 2] **Jasna Ilieva** [2] **Yoon Kim** [2]

Abstract

Current frontier LLMs can theoretically process long contexts with up to $1M$ tokens. But to what extent can they go beyond simple retrieval and perform deeper reasoning over such long contexts? We propose a simple synthetic benchmark to probe for long-horizon reasoning capabilities of LLMs, focusing on deductive logic expressed in Prolog. Our benchmark, dubbed PROLONG, systematically varies the complexity (reasoning depth) of problems, where the hardest case has a reasoning depth of 22 and $62k$ context length. We study 8 reasoning models across 5 families of frontier LLMs, and find that all models fail to perform above chance as the reasoning depth grows, with the majority failing beyond depth-10.

1. Introduction

Logical reasoning is central to tasks where reasoning models are increasingly expected to excel, such as in mathematical reasoning and coding. Large language models (LLMs) equipped with reasoning capabilities have shown impressive performance on a range of mathematical, logical, and coding tasks (OpenAI et al., 2024; Guo et al., 2025; Yang et al., 2025, inter alia). At what scale does the reasoning break down? We investigate this question through the lens of deductive logical reasoning, in which conclusions are derived necessarily from a set of premises by applying valid rules of inference.

We operationalize deductive reasoning as a query-answering problem over simple Prolog-style logic programs (Bratko, 2001), where the proof tree can be represented as a simple chain. As illustrated in Figure 1, each problem consists of a set of rules (implications) and facts (ground literals),

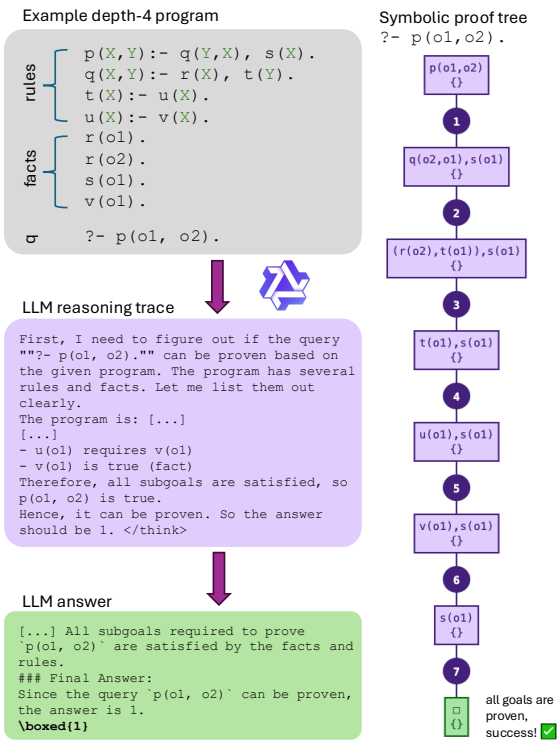

Figure 1. Example problem of depth-4, with an excerpt from the `Qwen-3-30B-Thinking` model[1](left). The symbolic derivation involves backward chaining through four rules until all subgoals are satisfied (right).

together with a query: can the query be proven given the program, or not? Answering correctly requires tracing a proof tree: starting from the query and recursively resolving each subgoal against the rules until all branches terminate in facts or fail. This process is *algorithmic*; a model that has learned a robust implementation of this algorithm should, in principle, generalize beyond the shallow depths seen in previous benchmarks (Clark et al., 2020; Tafjord et al., 2021; Saparov & He, 2023). In order to correctly answer the query, the model must correctly unify variables, follow argument orderings through rule heads and bodies (note the swapped arguments in "`p(X,Y) :- q(Y,X), s(X).`"), and track which subgoals have and have not been established. At greater depths, this process must be sustained across many steps and longer contexts.

[1]King Saud University, Riyadh, Saudi Arabia [2]Massachusetts Institute of Technology, Cambridge, United States of America. Correspondence to: Hadeel Al-Negheimish <hadeelan@mit.edu>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

[1]Full example in Appendix D

This paper systematically studies deductive reasoning with depth as a primary axis of complexity by constructing PROLONG, a benchmark of 2,400 logic programs in Prolog with reasoning depths up to 22. Crucially, this is also a long-horizon and long-context reasoning benchmark: context length scales as the number of rules and facts in the program grows with depth, reaching a median of $62k$ tokens. Evaluating these problems simultaneously stresses a model's ability to reason over extended chains and to locate and use relevant information within long contexts, two challenges that are typically studied in isolation. We perform a systematic evaluation and analysis of 8 frontier reasoning models, and find that even the best-performing frontier model, `GPT-5.5`, degrades to 67% accuracy at depth-20, with the majority of the models failing to perform above chance beyond depth-10. This reveals that even when logical inference is simple, frontier reasoning models struggle to maintain the required chain of substitutions and fact lookups as proof depth and context size scale.

In summary, our contributions are: (1) A framework for generating deductive reasoning benchmarks with precise control over depth and breadth; (2) The PROLONG benchmark, 2400 problems that span reasoning depths $0-22$ with contexts scaling up to $62k$ tokens; and (3) A systematic evaluation and analysis of 8 frontier reasoning models, revealing consistent and predictable failure patterns in long-context deductive reasoning.

2. Related Work

Early work on deductive reasoning benchmarks such as RuleTaker (Clark et al., 2020) and ProofWriter (Tafjord et al., 2021) established the basic setup of query answering over sets of rules and facts, with ProofWriter reaching a maximum proof depth of five steps. ProntoQA (Saparov & He, 2023) introduces proof-structure controls and formal error attribution, also evaluated up to five hops. These benchmarks revealed that models at the time struggle as proof depth increases, but the depths studied remain shallow. ZebraLogic (Lin et al., 2025) identifies sharp accuracy decline in constraint-satisfaction once the search space exceeds 10^7 possibilities. We show failure occurs in an even simpler setting: a smaller search space with a linear proof chain beyond depth 10. Another notable difference from previous work is that we opt to use Prolog instead of natural language in our evaluations, in order to avoid ambiguity and content effects that can impact performance, as models may draw on prior knowledge rather than the stated context (Lampinen et al., 2024).

On the long-context processing side, LongBench-v2 (Bai et al., 2024), RULER (Hsieh et al., 2024), and HELMET (Yen et al., 2025) evaluate retrieval and QA over extended contexts, treating reasoning and retrieval as independent axes in most tasks. PROLONG differs from these benchmarks in that every rule and fact in the program is necessary in the proof derivation, so the model cannot succeed by locating a few relevant pieces and ignoring the rest.

3. PROLONG: Prolog Long Reasoning

3.1. Background: Prolog

Prolog (Bratko, 2001) is a declarative programming language rooted in first-order logic. A Prolog program consists of a set of facts and rules, and an interpreter evaluates the truth value of a query given the context. A rule, for example "`p(X,Y) :- q(Y,X), s(X).`" from Figure 1, can be read as

$$\forall x \forall y \left((q(y,x) \wedge s(x)) \rightarrow p(x,y) \right)$$

which means that we can show $p(x,y)$ is true, if we can show that both $q(y,x)$ and $s(x)$ hold for any x, and y. The facts state unconditional truths, represented as relations between objects or object properties (predicates), ending with a period, e.g. "`r(o1).`". Variables are strings starting with an uppercase letter or underscore. An interpreter evaluates the truth value of a ground query, e.g. *"is `p(o1,o2)`. true?"* by goal-directed backward chaining through the context, and a query is determined to be true if all subgoals are satisfied, see example derivation in Figure 1 (right).

3.2. Data generation

The class of problems we consider in this framework is a smaller subset of definite Prolog logic programs, where the context can be represented as directed acyclic graphs (DAGs) (see Figure 2), with no loops or recursion in rule definitions. We also limit the arity of the predicates to be of maximum two. Furthermore, each predicate is defined with only one rule; this is intended to make the reasoning trace as simple as following a single chain, with no need for backtracking or navigating unsuccessful branches. The main building block of the data generation process is the GENERATERULE function (shown in Algorithm 1), which is called recursively to construct the context of each problem. We start by passing the main positive query, e.g. `p(o1,o2)`, which should be provable, along with the maximum depth and breadth, and the set of used predicate symbols.

The GENERATERULE function defines a rule with the target predicate as its head, and all the rules to define the body literals it includes up to the leaves. It also returns the minimum set of facts needed for the target query to succeed. An unprovable target is created by copying the set of facts, replacing them with the negative object tuple, and randomly mutating one of the facts by replacing one of the args with another constant such that the query is no longer provable. All rules and facts are then randomly shuffled. A single

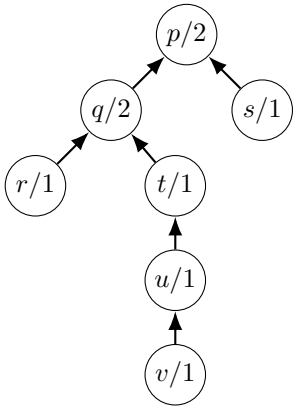

Figure 2. Dependency graph of the program example in Figure 1.

program (context) accompanies two samples in the data, one with a provable query and another unprovable, such that the dataset is balanced.

The algorithm by construction guarantees that at least one path in the DAG is as deep as length N, the other paths sampled uniformly up to that length. For each rule, we sample its breadth from a Poisson distribution with $\lambda = 2.5$, and a cut off according to $maxBreadth$. This is intended to skew the rules to having 2 body literals in order to control the size of the context. This framework allows systematic construction of random logic programs with controllable levels of complexity.

3.3. PROLONG Benchmark

We generate a set of 100 programs per depth, each with a provable and unprovable query, resulting in 200 samples per depth. We sample up to depth 22, skipping odd numbered N, in order to maximize coverage while minimizing cost. This results in a dataset of 2400 samples at varying levels of complexity, ranging from 133 median input tokens[2] at depth 0 to $62k$ tokens at depth 22. We verify that labels are correct with swi-prolog interpreter, which only takes half a second even for the largest programs. We create a canonical representation, *shape*, for the program to ensure context structural diversity, which is expectedly limited in depths 0 and 2. Dataset stats are shown in Table 1.

Algorithm 1 GENERATERULE(*head*, N, M, $\mathcal{P}_u$)

Require: ground literal *head*, depth $N \in \mathbb{Z}_{\geq 0}$, breadth $M \in \mathbb{Z}_{\geq 0}$, used predicate set $\mathcal{P}_u$
Ensure: Rules $\mathcal{R}$, facts $\mathcal{F}$

$\mathcal{R} \leftarrow \langle\rangle$; $\mathcal{F} \leftarrow \langle\rangle$
$U \leftarrow$ *head.args*
$\sigma \leftarrow \{U_i \mapsto x_i \mid i \in [|U|]\}$
$\widehat{head} \leftarrow$ LITERALWITHVARS(*head*, σ)
if $N = 0$ **then**
  $\mathcal{F}$.append(*head*)
  return $\mathcal{R}$, $\mathcal{F}$
end if
$\mathcal{B} \leftarrow \langle\rangle$
for $i \leftarrow 1$ **to** M **do**
  $\ell, \mathcal{P}'_u \leftarrow$ SAMPLELITERAL(U, $\mathcal{P}_u$)
  $N' \leftarrow \begin{cases} N-1 & \text{if } i = 1 \\ \mathcal{U}(0,\, N-1) & \text{otherwise} \end{cases}$
  $M' \leftarrow \text{Poisson}(\lambda, \textit{maxBreadth})$
  $(\mathcal{R}', \mathcal{F}') \leftarrow$ GENERATERULE(ℓ, N', M', $\mathcal{P}'_u$)
  $\mathcal{R} \mathrel{+}= \mathcal{R}'$; $\mathcal{F} \mathrel{+}= \mathcal{F}'$
  $\hat{\ell} \leftarrow$ LITERALWITHVARS(ℓ, σ)
  $\mathcal{B}$.append($\hat{\ell}$)
end for
Shuffle $\mathcal{B}$ uniformly at random
$\mathcal{R}$.append$\Big($*Rule*($\widehat{head}$, $\mathcal{B}$)$\Big)$
return $\mathcal{R}$, $\mathcal{F}$

4. Results

4.1. Experimental setup

Models We evaluated the benchmark on frontier reasoning models, both open-weight and proprietary: `Claude-Opus-4.7`, `GPT-5.5`, `GPT-5.4-mini`, `DeepSeek-V4-Pro`, `Kimi-K2.6`, `Qwen3.6-27B`, `Qwen3.5-27B`, and `Qwen3.5-4B`. Context sizes for these models range from $256k$ tokens to $1M$, full details are shown in Table 2 in the appendix. The models were sampled with setting the maximum output tokens to $64k$, except for `Qwen-3.5` and `3.6` models, where $82k$ tokens were recommended in the docs. When available, `reasoning effort` was set to high.

The system prompt asks the model to output the final answer $[0|1]$ in `\boxed{}`, which we extract and compare against the ground-truth.

4.2. How does depth affect performance?

Figure 3 presents accuracy as a function of reasoning depth for all evaluated models, with the median input token count annotated along the x-axis to highlight the simultaneous growth in context length. All models begin near ceiling performance at shallow depths, indicating that they are aware of the Prolog syntax. However, accuracy degrades monotonically as depth increases, with the rate and onset of decline varying considerably across models. Beyond depth 10, several models approach the chance baseline of 0.5 for this binary task. `GPT-5.5` is the clear outlier, maintaining high accuracy until depth 16, before declining at depths $20 - 22$. `Claude-Opus-4.7` is the second-best performer, remaining competitive with `GPT-5.5` through depth 12 before deteriorating sharply between depths 14 and 16, ultimately settling near chance at depth 22. The remaining models follow broadly similar trajectories of steeper decline. We emphasize that the context lengths are still well within the

[2]as measured by `DeepSeek-V4-Pro` tokenizer.

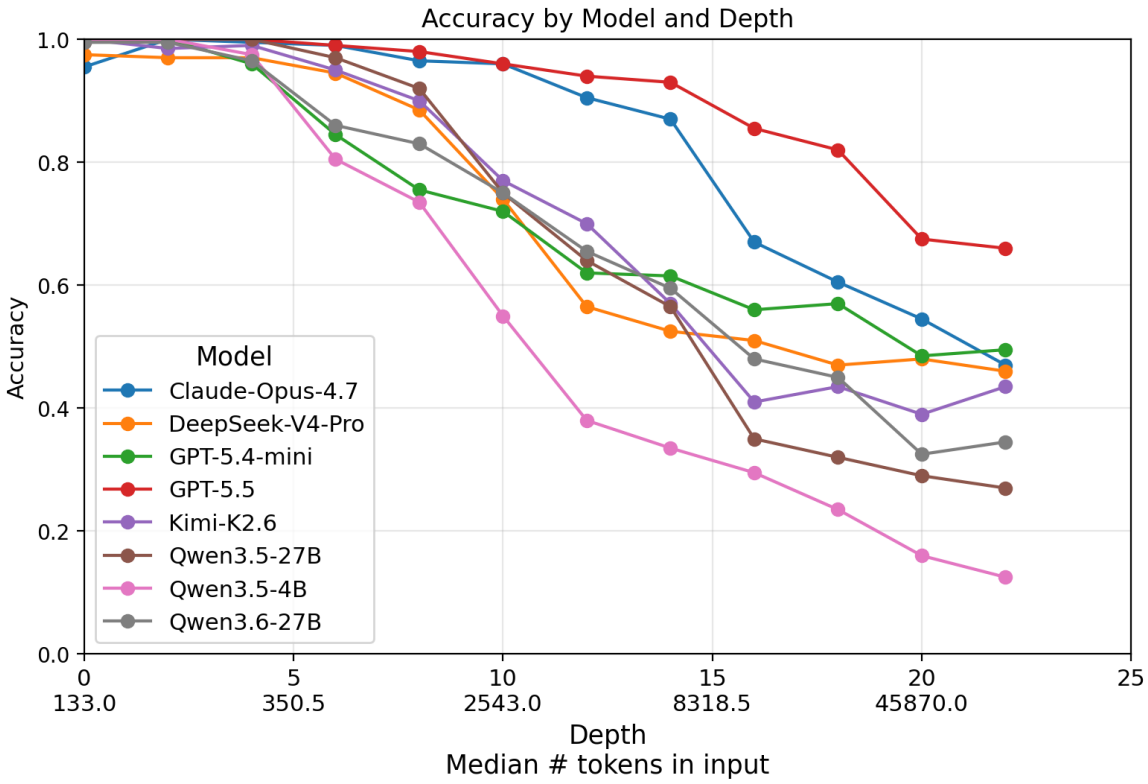

Figure 3. Model accuracy deteriorates with longer depths, with most models sharply declining after depth 10. Random chance is 0.5, since the dataset is balanced.

depth N	rules	facts	preds	unique shapes	tokens *(deepseek)*
0	0.0	2.0	1.0	2	133
2	3.0	9.0	7.0	77	197
4	9.0	26.0	21.0	100	351
6	23.0	66.0	56.5	100	690
8	51.0	138.0	118.5	100	1329
10	95.0	268.0	228.5	100	2543
12	202.5	544.0	474.5	100	4939
14	319.5	869.0	754.5	100	8319
16	666.0	1788.0	1556.5	100	17059
18	1052.5	2814.0	2457.5	100	26037
20	1820.0	4855.0	4255.0	100	45870
22	2424.5	6381.0	5604.0	100	62071

Table 1. Median PROLONG data statistics by depth.

advertised context lengths of these models.

4.3. Models become more *conservative*

We examine the frequency of final answer predictions in Figure 4. There appears to be a clear pattern in how models fail: while the benchmark is balanced by construction, with an equal number of provable/unprovable queries at every depth, many models exhibit a bias towards one answer, *unprovable* (red bars), that increases with the complexity of the program. We call this the *conservative* bias.

`Claude-Opus-4.7`, `DeepSeek-V4-Pro`, and `Kimi-K2.6` all develop a pronounced *conservative* bias beyond depth 10. By depth $16-18$, the proportion of negative predictions in these models dominates overwhelmingly. This suggests that models are effectively giving up, defaulting to "no" when the proof chain becomes too long to follow. `GPT-5.5` exhibits a milder version of the same tendency. `Qwen3.5` models' main failure mode is different: it runs out of tokens before reaching an answer, even at depth-10 where the median input size is only $2.5k$ tokens (grey bars).

4.4. Accuracy by provability

We show in Figure 6 how the accuracy is affected by depth divided by whether or not a query is provable. Given that many models tend to have a *conservative* bias, we find that half of the models maintain relatively high accuracies for unprovable queries regardless of depth. Whereas for provable queries, all models inevitably fail to find the appropriate proofs for at least half the queries.

4.5. Growth of reasoning trace with complexity

The growth of the reasoning trace varies across models, as shown in 5. `GPT-5.5`, `Claude-Opus-4.7`,

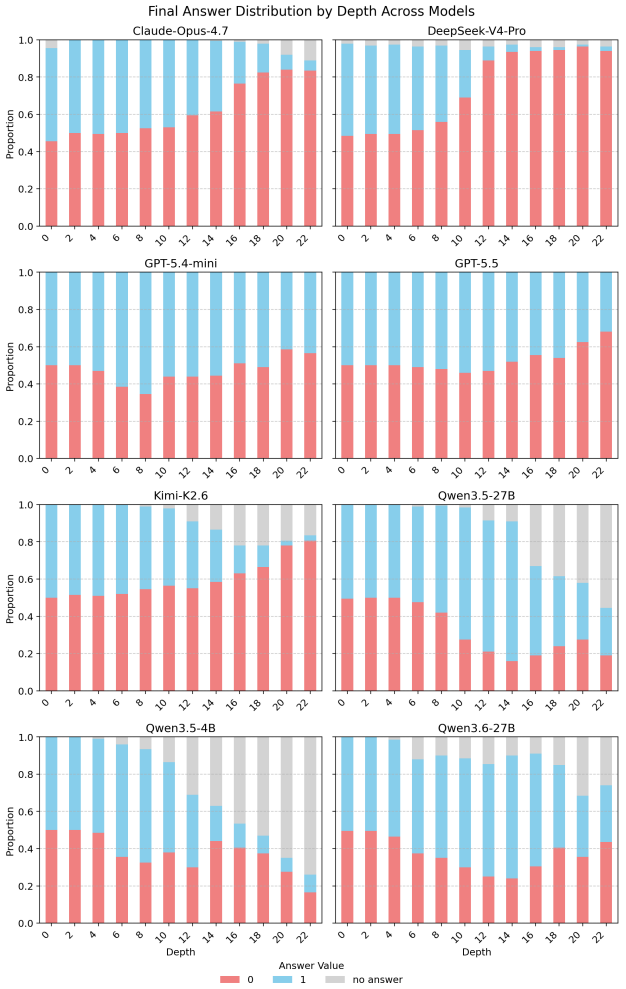

Figure 4. Proportion of extracted answers per model at each reasoning depth. Many models develop a pronounced *conservative* bias, predicting that a query is unprovable (red bars), even though the dataset is balanced.

`DeepSeek-V4-Pro`, and `GPT-5.4-mini` follow broadly similar trends, generating fewer than $20k$ output tokens across all depths with only modest increases as depth grows, suggesting these models allocate reasoning budget efficiently regardless of problem complexity. `Qwen3.6-27B` displays a notably irregular pattern: output length peaks around depths $8-10$ before declining to a plateau of approximately $40k$ tokens at greater depths (note that the maximum budget was $82k$). `Kimi-K2.6` similarly exhibits a non-monotonic profile, with output length peaking around depth 16 before declining. `Qwen3.5` stands apart from all other models in its tendency to allocate monotonically increasing reasoning tokens with depth, with output length continuing to grow even at the deepest levels, a pattern consistent with the model expanding its thinking trace without necessarily improving its derivations.

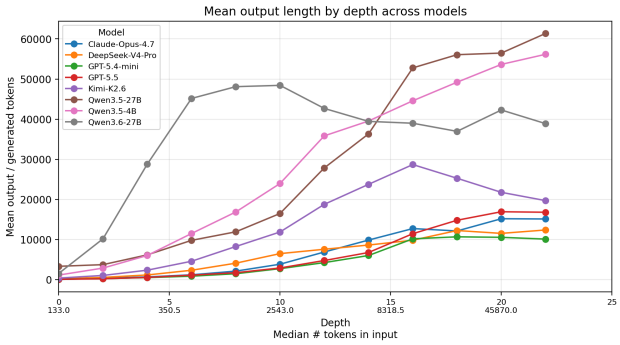

Figure 5. Mean number of output tokens per model. Only `Qwen-3.5` continues to grow monotonically, with the rest remaining significantly below the token budget.

4.6. What does the trace say?

We investigate the reasoning trace at depth 22 of the best-performing open-source model, `DeepSeek-V4-Pro`, and the reasoning summary provided by `GPT-5.5` (best overall, but the entire trace is kept private). We analyze the reasoning traces of False Negatives and True Negatives. The former would tell us why it failed even when evidence to the contrary exists, and the latter would inform us whether the model correctly identified the missing fact or was right for the wrong reasons. Analyzing (with an LLM—`GPT-5.5`—and additionally verifying manually) the reasoning summary produced by `GPT-5.5`, we find that in the False Negatives, the model starts the reasoning chain correctly, but then "loses its way". It claims that a necessary fact is missing or has no rules to support it, even though it exists in the context. For the True Negatives, LLM analysis indicates that of the 84 True Negatives, only 34 actually cited the correct missing fact (40.5%), and the vast majority did not (59.5%). The other cases were: 14 only returned the final answer, without reasoning, 15 give a vague and incomplete proof, and 21 mention other facts as blockers even though they are provable. This indicates that the reasoning is not faithful to the actual derivation of a query. These patterns are even more detrimental in `DeepSeek-V4-Pro` traces, where 97 provable queries were falsely claimed to be unsupported by arbitrary conditions that exist in the context. As for the 91 true negatives, only 11% of their reasoning traces mention the planted missing fact correctly.

5. Conclusion

We present a framework and a benchmark for studying deductive reasoning at scale, controlling complexity with maximum depth of reasoning. Our evaluations on frontier large reasoning models show that while determining the truth value of queries is consistent in smaller programs, models deteriorate as the context scales.

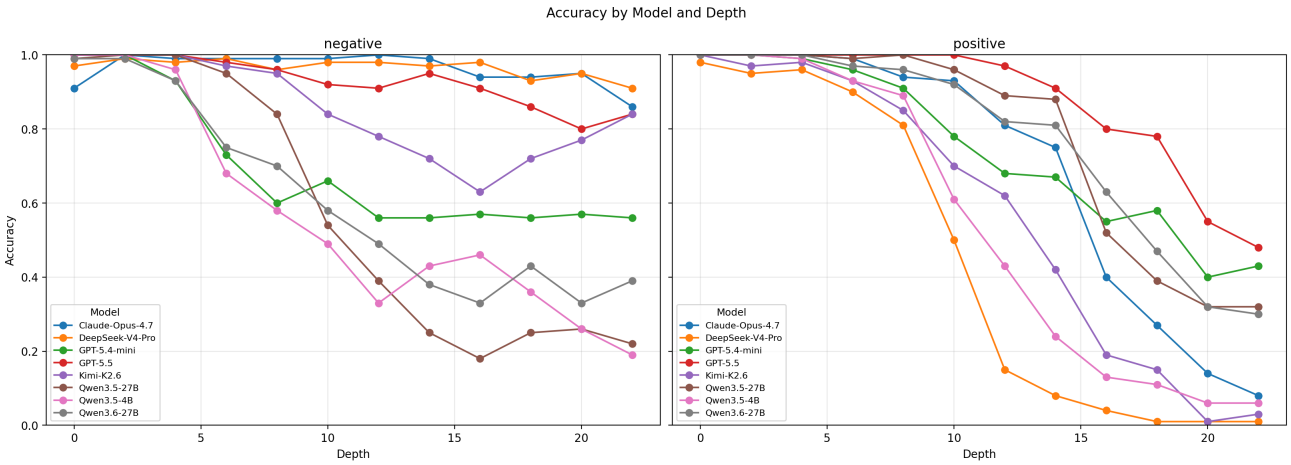

Figure 6. Accuracy per target provability. The deterioration in performance is sharper for provable queries as complexity increases, which aligns with the models' *conservative* bias.

References

Bai, Y., Tu, S., Zhang, J., Peng, H., Wang, X., Lv, X., Cao, S., Xu, J., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Bratko, I. *Prolog programming for artificial intelligence*. International computer science series. Addison Wesley, Harlow, England, 3rd ed. edition, 2001. ISBN 0201403757.

Clark, P., Tafjord, O., and Richardson, K. Transformers as soft reasoners over language. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/537. URL https://doi.org/10.24963/ijcai.2020/537. Main track.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., and Ginsburg, B. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Lampinen, A. K., Dasgupta, I., Chan, S. C. Y., Sheahan, H. R., Creswell, A., Kumaran, D., McClelland, J. L., and Hill, F. Language models, like humans, show content effects on reasoning tasks. *PNAS Nexus*, 3(7):pgae233, 07 2024. ISSN 2752-6542. doi: 10.1093/pnasnexus/pgae233. URL https://doi.org/10.1093/pnasnexus/pgae233.

Lin, B. Y., Bras, R. L., Richardson, K., Sabharwal, A., Poovendran, R., Clark, P., and Choi, Y. Zebralogic: on the

scaling limits of llms for logical reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

OpenAI, :, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., Iftimie, A., Karpenko, A., Passos, A. T., Neitz, A., Prokofiev, A., Wei, A., Tam, A., Bennett, A., Kumar, A., Saraiva, A., Vallone, A., Duberstein, A., Kondrich, A., Mishchenko, A., Applebaum, A., Jiang, A., Nair, A., Zoph, B., Ghorbani, B., Zhang, B., Rossen, B., Sokolowsky, B., Barak, B., McGrew, B., Minaiev, B., Hao, B., Baker, B., Houghton, B., McKinzie, B., Eastman, B., Lugaresi, C., Bassin, C., Hudson, C., Li, C. M., de Bourcy, C., Voss, C., Shen, C., Zhang, C., Koch, C., Orsinger, C., Hesse, C., Fischer, C., Chan, C., Roberts, D., Kappler, D., Levy, D., Selsam, D., Dohan, D., Farhi, D., Mely, D., Robinson, D., Tsipras, D., Li, D., Oprica, D., Freeman, E., Zhang, E., Wong, E., Proehl, E., Cheung, E., Mitchell, E., Wallace, E., Ritter, E., Mays, E., Wang, F., Such, F. P., Raso, F., Leoni, F., Tsimpourlas, F., Song, F., von Lohmann, F., Sulit, F., Salmon, G., Parascandolo, G., Chabot, G., Zhao, G., Brockman, G., Leclerc, G., Salman, H., Bao, H., Sheng, H., Andrin, H., Bagherinezhad, H., Ren, H., Lightman, H., Chung, H. W., Kivlichan, I., O'Connell, I., Osband, I., Gilaberte, I. C., Akkaya, I., Kostrikov, I., Sutskever, I., Kofman, I., Pachocki, J., Lennon, J., Wei, J., Harb, J., Twore, J., Feng, J., Yu, J., Weng, J., Tang, J., Yu, J., Candela, J. Q., Palermo, J., Parish, J., Heidecke, J., Hallman, J., Rizzo, J., Gordon, J., Uesato, J., Ward, J., Huizinga, J., Wang, J., Chen, K., Xiao, K., Singhal, K., Nguyen, K., Cobbe, K., Shi, K., Wood, K., Rimbach, K., Gu-Lemberg, K., Liu, K., Lu, K., Stone, K., Yu, K., Ahmad, L., Yang, L., Liu, L., Maksin, L., Ho, L., Fedus, L., Weng, L., Li, L., McCallum, L., Held, L., Kuhn, L., Kondraciuk, L., Kaiser, L., Metz, L., Boyd, M., Trebacz, M., Joglekar, M., Chen, M., Tintor, M., Meyer, M., Jones, M., Kaufer, M., Schwarzer, M., Shah, M., Yatbaz, M., Guan, M. Y., Xu, M., Yan, M., Glaese, M., Chen, M., Lampe, M., Malek, M., Wang, M., Fradin, M., McClay, M., Pavlov, M., Wang, M., Wang, M., Murati, M., Bavarian, M., Rohaninejad, M., McAleese, N., Chowdhury, N., Chowdhury, N., Ryder, N., Tezak, N., Brown, N., Nachum, O., Boiko, O., Murk, O., Watkins, O., Chao, P., Ashbourne, P., Izmailov, P., Zhokhov, P., Dias, R., Arora, R., Lin, R., Lopes, R. G., Gaon, R., Miyara, R., Leike, R., Hwang, R., Garg, R., Brown, R., James, R., Shu, R., Cheu, R., Greene, R., Jain, S., Altman, S., Toizer, S., Toyer, S., Miserendino, S., Agarwal, S., Hernandez, S., Baker, S., McKinney, S., Yan, S., Zhao, S., Hu, S., Santurkar, S., Chaudhuri, S. R., Zhang, S., Fu, S., Papay, S., Lin, S., Balaji, S., Sanjeev, S., Sidor, S., Broda, T., Clark, A., Wang, T., Gordon, T., Sanders, T., Patwardhan, T., Sottiaux, T., Degry, T., Dimson, T., Zheng, T., Garipov, T., Stasi, T., Bansal, T., Creech, T., Peterson, T., Eloundou, T., Qi, V., Kosaraju, V., Monaco, V., Pong, V., Fomenko, V., Zheng, W., Zhou, W., Zhan, W., McCabe, W., Zaremba, W., Dubois, Y., Lu, Y., Chen, Y., Cha, Y., Bai, Y., He, Y., Zhang, Y., Wang, Y., Shao, Z., and Li, Z. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Saparov, A. and He, H. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Tafjord, O., Dalvi, B., and Clark, P. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yen, H., Gao, T., Hou, M., Ding, K., Fleischer, D., Izsak, P., Wasserblat, M., and Chen, D. Helmet: How to evaluate long-context language models effectively and thoroughly. In *International Conference on Learning Representations (ICLR)*, 2025.

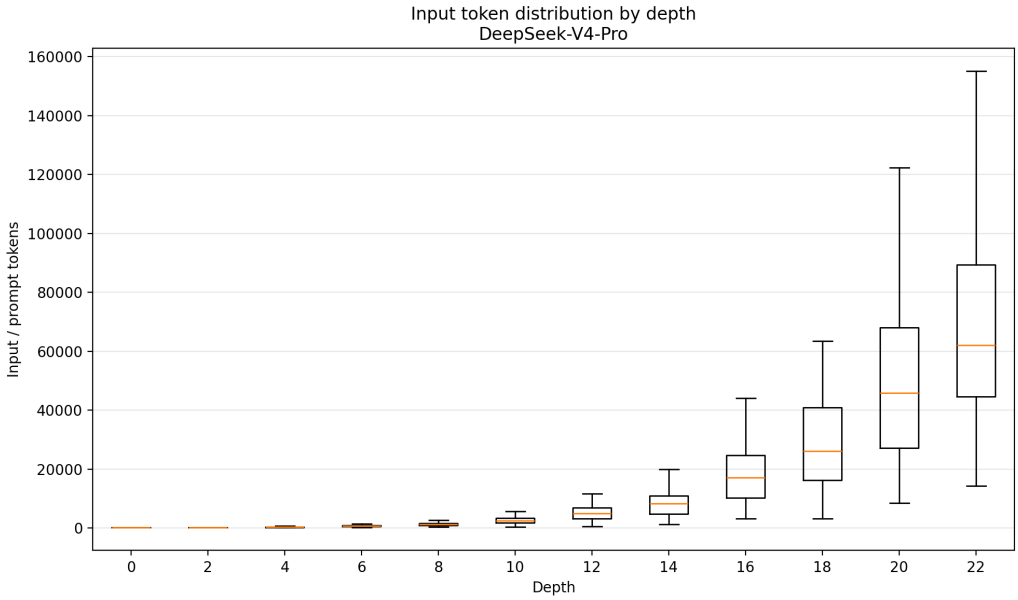

Figure 7. Boxplots for input tokens per Depth for `DeepSeek-V4-Pro`

A. Models

We evaluated frontier reasoning models, both open-weight and proprietary: `Claude-Opus-4.7`, `GPT-5.5`, `GPT-5.4-mini`, `DeepSeek-V4-Pro`, `Kimi-K2.6`, `Qwen3.6-27B`, `Qwen3.5-27B`, and `Qwen3.5-4B`. Full details are in Table 2.

Table 2. Comparison of frontier models used in this analysis based on publicly available data.

Model name	Weights	Model size	Release date	Context size
`Claude-Opus-4.7`	🔒	Not disclosed	Apr. 16, 2026	$1M$ tokens
`GPT-5.5`	🔒	Not disclosed	Apr. 23, 2026	$1M$ tokens
`GPT-5.4-mini`	🔒	Not disclosed	Mar. 17, 2026	$400k$ tokens
`DeepSeek-V4-Pro`	🔑	$1.6T$ ($49B$ activated)	Apr. 24, 2026	$1M$ tokens
`Kimi-K2.6`	🔑	$1T$ ($32B$ activated)	Apr. 20, 2026	$256k$ tokens
`Qwen3.6-27B`	🔑	$27B$	Apr. 21, 2026	$262k$ native; up to $\sim 1M$ extended
`Qwen3.5-27B`	🔑	$27B$	Feb. 24, 2026	$262k$ native; up to $\sim 1M$ extended
`Qwen3.5-4B`	🔑	$4B$	Feb. 27, 2026	$262k$ native; up to $\sim 1M$ extended

B. PROLONG Data

The size of the context grows significantly with depth, Figure 7 shows boxplots of the input tokens as measured by `DeepSeek-V4-Pro`. Note that they are similar across the board, with a slight increase in `Claude-Opus-4.7`. Figure 2 shows the context DAG for the example in Figure 1.

We analyze the proofs derived by a symbolic solver, SWI-Prolog. Tables 3 and 4 show the median time and reasoning trace of SWI-Prolog for the provable and unprovable queries by depth.

depth	n_examples	time (s)	trace_len
0	100	0.03	3.0
2	100	0.03	21.0
4	100	0.03	63.0
6	100	0.03	169.5
8	100	0.04	355.5
10	100	0.05	685.5
12	100	0.07	1423.5
14	100	0.08	2263.5
16	100	0.13	4669.5
18	100	0.19	7372.5
20	100	0.34	12765.0
22	100	0.42	16812.0

Table 3. Median time and trace length by depth for provable queries.

depth	n_examples	time (s)	trace_len
0	100	0.03	2.0
2	100	0.03	14.0
4	100	0.03	32.0
6	100	0.03	80.0
8	100	0.04	177.5
10	100	0.05	353.0
12	100	0.07	635.0
14	100	0.08	920.0
16	100	0.12	1952.0
18	100	0.18	3278.0
20	100	0.31	4412.0
22	100	0.41	6183.5

Table 4. Median time and trace length by depth for unprovable queries.

C. Prompt Template

System Prompt

```
You are a helpful and harmless assistant. Consider the following prolog-like
declarative logic program, arguments starting with capital letters represent
variables, lowercase letters represent constants. \+ is a symbol that represents
negation by failure, a negated literal in a rule is satisfied when it fails to be
proven. Decide if a query can be proven given the background program and return 1,
otherwise return 0 if it cannot be proven. A query is the final line starting with
""?- "". You should think step-by-step. Put your final answer within \\boxed{{}}.
```

D. Full Example

Qwen-3-Thinking Input

```
<|im_start|>system
You are a helpful and harmless assistant. Consider the following prolog-like
declarative logic program, arguments starting with capital letters represent
variables, lowercase letters represent constants. \+ is a symbol that represents
negation by failure, a negated literal in a rule is satisfied when it fails to be
```

```
proven. Decide if a query can be proven given the background program and return 1,
otherwise return 0 if it cannot be proven. A query is the final line starting with
""?- "". You should think step-by-step. Put your final answer within \\boxed{{}}.<|
im_end|>
<|im_start|>user
p(X,Y):- q(Y,X), s(X).
q(X,Y):- r(X), t(Y).
t(X):- u(X).
u(X):- v(X).
r(o1).
r(o2).
s(o1).
v(o1).
?- p(o1, o2).<|im_end|>
<|im_start|>assistant
<think>
```

Qwen-3-30B Thinking Reasoning Content

```
First, I need to figure out if the query ""?- p(o1, o2)."" can be proven based on
the given program. The program has several rules and facts. Let me list them out
clearly.

The program is:

- p(X,Y) :- q(Y,X), s(X).

- q(X,Y) :- r(X), t(Y).

- t(X) :- u(X).

- u(X) :- v(X).

- r(o1).

- r(o2).

- s(o1).

- v(o1).

The query is p(o1, o2).

I need to see if p(o1, o2) can be proven using these rules. Since it's a declarative
 logic program, I should try to derive it step by step.

Start with the query: p(o1, o2).

From the first rule, p(X,Y) is true if q(Y,X) and s(X) are both true.

So for p(o1, o2), X is o1 and Y is o2, so it requires q(o2, o1) and s(o1).

Now, s(o1) is given as a fact: s(o1). So s(o1) is true. That part is satisfied.

Now, I need to check if q(o2, o1) can be proven.

q(X,Y) is true if r(X) and t(Y) are both true.

For q(o2, o1), X is o2, Y is o1, so it requires r(o2) and t(o1).
```

```
r(o2) is given as a fact: r(o2). So r(o2) is true.

Now, I need t(o1).

t(X) is true if u(X) is true.

u(X) is true if v(X) is true.

v(o1) is given as a fact: v(o1). So v(o1) is true.

Therefore, u(o1) should be true because u(X) :- v(X), so with X=o1, u(o1) is true.

Then t(o1) should be true because t(X) :- u(X), so with X=o1, t(o1) is true.

Therefore, t(o1) is true.

So for q(o2, o1), we have r(o2) true and t(o1) true, so q(o2, o1) is true.

Then, for p(o1, o2), we have q(o2, o1) true and s(o1) true, so p(o1, o2) should be
true.

Let me double-check.

List all the steps:

1. Query: p(o1, o2)

2. From p(X,Y) :- q(Y,X), s(X), so for X=o1, Y=o2: need q(o2, o1) and s(o1)

3. s(o1) is fact, so s(o1) = true

4. Now need q(o2, o1)

5. From q(X,Y) :- r(X), t(Y), so for X=o2, Y=o1: need r(o2) and t(o1)

6. r(o2) is fact, so r(o2) = true

7. Now need t(o1)

8. From t(X) :- u(X), so for X=o1: need u(o1)

9. From u(X) :- v(X), so for X=o1: need v(o1)

10. v(o1) is fact, so v(o1) = true

11. Therefore u(o1) = true (from u(X):-v(X))

12. Therefore t(o1) = true (from t(X):-u(X))

13. Therefore q(o2, o1) = true (from q(X,Y):-r(X),t(Y) with X=o2,Y=o1)

14. Therefore p(o1, o2) = true (from p(X,Y):-q(Y,X),s(X) with X=o1,Y=o2)

All steps seem to check out. But let me make sure there are no negations or anything
 that might cause failure.

The program has \+ for negation by failure, but in this case, there are no negated
literals in the rules or the query. The rules are all positive implications.

For example, the rule for p has q(Y,X) and s(X), both positive.

q has r(X) and t(Y), positive.
```

```
t has u(X), positive.

u has v(X), positive.

Facts are given directly.

So no negation to worry about.

Also, the variables: in the query, o1 and o2 are constants, not variables, so we're
directly substituting.

In the rule, X and Y are variables, but in the query, we're using specific constants.


So it should be fine.

I should confirm if all the facts are present.

r(o1) and r(o2) are both facts, so r is true for o1 and o2.

s(o1) is fact, so s is true for o1.

v(o1) is fact, so v is true for o1.

Now, for t(o1), we need u(o1), which needs v(o1), which is true.

Similarly, for q(o2,o1), we need r(o2) which is true, and t(o1) which we just
established.

r(o2) is true, yes.

Is there any other path? I think this is the only way.

The query is p(o1,o2), and we've derived it.

But let me see the rule for p: p(X,Y) :- q(Y,X), s(X).

So for p(o1,o2), it requires q(Y,X) with Y=o2, X=o1, so q(o2,o1).

Yes.

In q(X,Y), it's r(X) and t(Y), so for q(o2,o1), X=o2, Y=o1, so r(o2) and t(o1).

All good.

I could think in terms of the body.

The body of p(o1,o2) is q(o2,o1) and s(o1).

s(o1) is true.

q(o2,o1) requires r(o2) and t(o1).

r(o2) is true.

t(o1) requires u(o1), which requires v(o1), which is true.

So yes.

The program also has r(o1), but we didn't use it for this query, which is fine.

Similarly, s(o1) is used, but s(o2) isn't given, so we don't care.
```

```
v(o1) is used, but v(o2) isn't given, so not needed.

So for this specific query, it should be provable.

Therefore, the answer should be 1.

The question says ""return 1 if it can be proven, otherwise 0"".

So I think it can be proven.

But let me make sure about the negation by failure. The problem mentions that \+
represents negation by failure, but in this program, there is no \+ used. So no
issue.

For example, if there was a rule like p(X) :- \+ q(X), but here there isn't.

So all good.

Perhaps I should write it out as a proof tree.

Query: p(o1, o2)

- p(o1, o2) requires q(o2, o1) and s(o1)

- s(o1) is true (fact)

- q(o2, o1) requires r(o2) and t(o1)

- r(o2) is true (fact)

- t(o1) requires u(o1)

- u(o1) requires v(o1)

- v(o1) is true (fact)

Therefore, all subgoals are satisfied, so p(o1, o2) is true.

Hence, it can be proven.

So the answer should be 1.

The problem says ""arguments starting with capital letters represent variables,
lowercase letters represent constants."" Here, o1 and o2 are lowercase, so constants,
 which is fine.

In the rule, X and Y are variables, but we're instantiating them.

No issue.

I think I'm confident.
</think>
```

Qwen-3-30B Thinking Answer

```
To determine if the query `?- p(o1, o2).` can be proven based on the given program,
we need to check if the rules and facts allow us to derive `p(o1, o2)` through
logical inference. The program is declarative, and negation by failure (`\+`) is not
 present in any rules or the query, so we only deal with positive literals.
```

```
### Step-by-Step Derivation:
1. **Query: `p(o1, o2)`**
   - From the rule `p(X, Y) :- q(Y, X), s(X)`, we substitute `X = o1` and `Y = o2`:
     - This requires `q(o2, o1)` and `s(o1)` to be true.

2. **Check `s(o1)`**:
   - `s(o1)` is given as a fact in the program: `s(o1).`
   - Therefore, `s(o1)` is true.

3. **Check `q(o2, o1)`**:
   - From the rule `q(X, Y) :- r(X), t(Y)`, we substitute `X = o2` and `Y = o1`:
     - This requires `r(o2)` and `t(o1)` to be true.

4. **Check `r(o2)`**:
   - `r(o2)` is given as a fact in the program: `r(o2).`
   - Therefore, `r(o2)` is true.

5. **Check `t(o1)`**:
   - From the rule `t(X) :- u(X)`, we substitute `X = o1`:
     - This requires `u(o1)` to be true.
   - From the rule `u(X) :- v(X)`, we substitute `X = o1`:
     - This requires `v(o1)` to be true.
   - `v(o1)` is given as a fact in the program: `v(o1).`
     - Therefore, `v(o1)` is true.
   - Since `v(o1)` is true, `u(o1)` is true (from `u(X) :- v(X)`).
   - Since `u(o1)` is true, `t(o1)` is true (from `t(X) :- u(X)`).

6. **Conclusion for `q(o2, o1)`**:
   - `r(o2)` is true and `t(o1)` is true, so `q(o2, o1)` is true.

7. **Conclusion for `p(o1, o2)`**:
   - `q(o2, o1)` is true and `s(o1)` is true, so `p(o1, o2)` is true.

All subgoals required to prove `p(o1, o2)` are satisfied by the facts and rules.
There are no negated literals or failures that prevent the proof.

### Final Answer:
Since the query `p(o1, o2)` can be proven, the answer is 1.
\boxed{1}
```