EFFICIENT DEDUCTIVE NATURAL LANGUAGE REASONING WITH A*-INFORMED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many applications of large language models (LLMs) require deductive reasoning, yet models frequently produce incorrect or redundant inference steps. We frame natural language inference as search in a hypergraph, where a correct solution requires a valid proof rather than just a final answer. Under this framework, we investigate whether LLMs can learn to generate correct proofs in an efficient manner with guidance from forward chaining algorithms, specifically A* search. We explore two training techniques: supervised fine-tuning on execution traces from A* and reinforcement learning with A*-informed reward models. Empirically, we find that Llama-3.2 models in the 1B–3B range benefit substantially from A*-guided training, going from near-zero accuracy to outperforming DeepSeek-V3.2. Our analysis uncovers a trade-off: while correctness rewards maximize accuracy, A*-informed signals strike a balance between accuracy and efficiency. Furthermore, we find that on larger search spaces, models trained with imperfect heuristics exhibit superior accuracy to learning from a true cost-to-go oracle. Our results demonstrate a promising direction towards reasoning guided by classical search algorithms.

1 INTRODUCTION

Reasoning has emerged as one of the most prominent use cases of modern large language models (LLMs). Chain-of-thought prompting (Wei et al., 2022) first demonstrated that providing user-specified reasoning templates can dramatically improve LLM performance. Subsequently, the field shifted towards training models to produce such reasoning templates autonomously, most recently by means of reinforcement learning with verifiable rewards (RLVR; Guo et al., 2025). Models trained under this paradigm have achieved striking results on reasoning-based tasks such as solving competition mathematics (Luong et al., 2025) and graduate-level science questions (Rein et al., 2024).

Yet, LLMs remain highly flawed reasoners in several respects: While they may often arrive at the correct final answer, they frequently generate reasoning chains that do not faithfully reflect the process by which they arrive at that answer (Turpin et al., 2023; Arcuschin et al., 2025), show sensitivity to perturbations of a problem's surface-level features (Jiang et al., 2024; Mirzadeh et al., 2025), fail on rigorous proof writing (Mahdavi et al., 2025; Petrov et al., 2025), struggle to generalize to complex problems (Dziri et al., 2023; Kordi et al., 2026), and exhibit human-like reasoning biases (Lampinen et al., 2024). Beyond correctness, another pressing concern is *efficiency*: models routinely generate reasoning chains that contain unnecessary inference steps (Chen et al., 2025; Opedal et al., 2025). This matters practically—such redundant detours reduce interpretability and undermine the trust that a human evaluator would place in an agent's proof. It also matters scientifically, since inefficient reasoning poses a scalability barrier as task complexity increases (cf. Lawsen, 2025).

The aforementioned shortcomings are unsurprising given the dominant paradigm of training LLMs to maximize rewards based solely on final-answer correctness (Shao et al., 2024), which provides no learning signal for the correctness or efficiency of intermediate steps. A promising remedy is to train the model to not only generate correct answers, but to learn how to *search* to find such answers (Gandhi et al., 2024; 2025; Lehnert et al., 2024; Sel et al., 2024). In this paper, we study whether LLMs can learn the search process underlying **deductive reasoning**, expressing their inferences in natural language. We formalize deductive reasoning as search in a hypergraph à la Klein & Manning (2001), where vertices represent theorems and hyperedges correspond to valid deductions under a logic program. The LLM must efficiently navigate this hypergraph, finding a path from a set of start

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

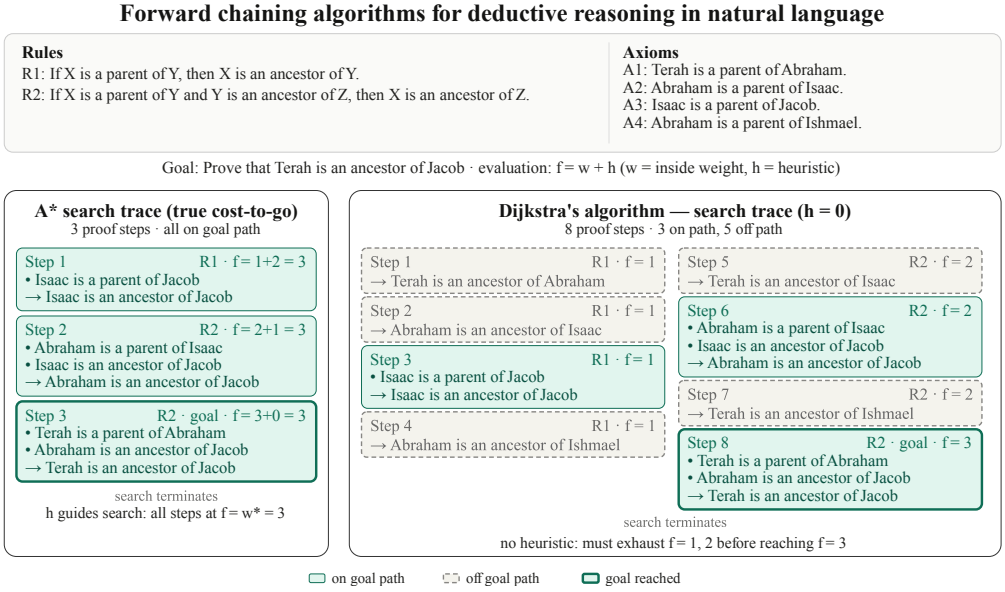

Figure 1: We study whether LLMs can learn to reason correctly and efficiently in natural language under guidance from A* search. A* can yield efficient search in practice, as shown in this illustration.

vertices (axioms) to a goal vertex—all expressed as a proof in natural language. Importantly, since any error renders a proof incorrect, we only credit a candidate proof if all inferences required to prove the goal are correct.

To derive learning signals for this search problem, we consult well-known forward-chaining procedures, specifically, A* search (Hart et al., 1968). A* is a classical path-finding algorithm which, when provided with an appropriate heuristic estimate of the cost-to-go, yields optimally efficient search (Dechter & Pearl, 1985) with strong empirical performance on planning tasks (Russell & Norvig, 2021). We consider three heuristics: an imperfect heuristic derived from a relaxed version of the search graph, an oracle providing the true cost-to-go (yielding the shortest proof with no search; fn 4), and the zero heuristic (i.e., Dijkstra's (1959) algorithm). Employing signals from our A* implementation, we train LLMs to efficiently generate correct proofs in natural language. We use two training methods: (1) imitation of search traces generated by execution of A* using supervised fine-tuning (SFT) and (2) reinforcement learning (RL) with the GRPO objective (Shao et al., 2024), guided by reward signals derived from the priority function in the A* algorithm.

We train models in the 1B–3B range and find that they benefit substantially from A* learning signals. For example, Llama-3.2-3B-Instruct goes from $\approx 0\%$ accuracy to 93.5% accuracy and 97.5% efficiency on ProofWriter (Tafjord et al., 2021), surpassing DeepSeek-V3.2 with in-context learning. We find the role of search heuristic to be data dependent: On ProofWriter data, which has a relatively small search space, there is no significant difference between the imperfect heuristic and the shortest proof from the true cost-to-go oracle. On a second dataset with a larger search space (Rameshkumar et al., 2025), however, the model fails on most of the held-out data when trained under the true cost-to-go oracle, while the imperfect heuristic yields high scores. We consider various reward models: a simple correctness reward, an efficiency reward based on step counts, and A*-informed rewards. In so doing, we uncover a trade-off between accuracy and efficiency—all reward models appear on the accuracy-efficiency Pareto front. Overall, our paper demonstrates a promising path toward using guidance of classical planning algorithms to learn correct and efficient proof writing in natural language.

2 Background on Deductive Reasoning and A* Search

We provide background on deductive reasoning under logic programs (§2.1) and the A* search algorithm (§2.2). A* was originally conceived as an algorithm over standard weighted graphs; however, we require its extension to *hyper*graphs (Felzenszwalb & McAllester, 2007) in order to account for the fact that inference rules may have more than one premise, such as R2 in Fig. 1.

2.1 Deductive Reasoning Formalized as Logic Programming

We consider a notion of logic programming based on Datalog (Ceri et al., 1989), which captures a broad class of deductive reasoning tasks computable in polynomial time (Vardi, 1982; Immerman, 1986). Logic programming is an established framework for studying LLM reasoning (e.g., Clark et al., 2021; Zhang et al., 2023). This subsection introduces the necessary background and notation.

An **item** (or **atom**) is an expression of the form $\mathrm{p}(t_1, ..., t_N)$, where p is a **predicate** and the arguments $t_1, ..., t_N \in \Sigma_x \cup \Sigma_X$ are **terms**. Terms can be constants (**ground** terms), denoted x, y, z, or variables (**nonground** terms), denoted X, Y, Z. An **inference rule** is an expression of the form $b_1, ..., b_K \vdash c$, where $b_1, ..., b_K, c$ are items; $b_1, ..., b_K$ are called **premises** and c is the **conclusion**. For example,

$$\mathrm{parent}(X, Y),\ \mathrm{ancestor}(Y, Z) \vdash \mathrm{ancestor}(X, Z)$$

is an inference rule that allows us to conclude that if X is a parent of Y and Y is an ancestor of Z, then X is an ancestor of Z. We assume that the inference rules are **range restricted**, meaning that each variable appearing in the conclusion c also appears in at least one item b_k in the premise. Inference rules with a null premise, i.e., where $K = 0$, and a ground conclusion are called **axioms**. For example, $\mathrm{parent}(\mathtt{abraham}, \mathtt{isaac})$, omitting the $\vdash$ symbol, is an axiom. A **logic program** $\mathcal{P}$ is simply a set of inference rules, often written $\mathcal{P} = \mathcal{R} \sqcup A$ to separate axioms A from other rules $\mathcal{R}$. For example,

$$\mathrm{parent}(\mathtt{terah}, \mathtt{abraham}) \quad \mathrm{parent}(\mathtt{abraham}, \mathtt{ishmael})$$
$$\mathrm{parent}(\mathtt{abraham}, \mathtt{isaac}) \quad \mathrm{parent}(\mathtt{isaac}, \mathtt{jacob})$$
$$\mathrm{parent}(X, Y) \vdash \mathrm{ancestor}(X, Y) \quad \mathrm{parent}(X, Y),\ \mathrm{ancestor}(Y, Z) \vdash \mathrm{ancestor}(X, Z)$$

is a simple logic program that expresses biblical ancestries (Sterling & Shapiro, 1994). A program $\mathcal{P}$ induces a **Herbrand base** H, which is the set of all items that can be formed by predicates and ground terms that occur in $\mathcal{P}$. A subset of the Herbrand base $I \subseteq H$ is called an **interpretation**.

We require some additional definitions to assign truth values to items. A **substitution** θ is a finite set of pairs $\{(X_m, t_m)\}_{m=1}^{M}$, where $X_m \in \Sigma_X$, $t_m \in \Sigma_x \cup \Sigma_X$, $X_m \neq X_{m'}$ for all $m \neq m'$, and $X_m \neq t_m$ for all m. Substitutions θ can be applied to items b, e.g., $\mathrm{parent}(X, \mathtt{isaac})/\{(X, \mathtt{abraham})\} = \mathrm{parent}(\mathtt{abraham}, \mathtt{isaac})$. Let $\Theta(\mathcal{P})$ denote the set of all substitutions under $\mathcal{P}$. We can write $b'_1, ..., b'_K \vdash c'$ if there exists a substitution $\theta \in \Theta(\mathcal{P})$ such that $b'_1, ..., b'_K \vdash c' = b_1/\theta, ..., b_K/\theta \vdash c/\theta$; this is called an **instantiation** of $b_1, ..., b_K \vdash c$. A **ground instantiation** is an instantiation that does not contain variables. We will generally use $b'_1, ..., b'_K, c'$ as shorthand for the items in ground instantiations of $b_1, ..., b_K \vdash c$. Now, a program $\mathcal{P}$'s **fixpoint operator** $\mathrm{T}_{\mathcal{P}} \colon 2^H \to 2^H$ is defined as

$$\mathrm{T}_{\mathcal{P}}(I) = I \cup (\{c/\theta \mid (b_1, ..., b_K \vdash c) \in \mathcal{P},\ \wedge_{k=1}^{K} b_k/\theta \in I,\ \theta \in \Theta(\mathcal{P})\} \cap H), \tag{1}$$

where $I \subseteq H$ is an interpretation. This operator allows us to assign semantics: The **minimal Herbrand model** is $M = \mathrm{T}^*_{\mathcal{P}}(A) \stackrel{\text{def}}{=} \bigcup_{n=0}^{\infty} \mathrm{T}^n_{\mathcal{P}}(A)$, where $\mathrm{T}^n_{\mathcal{P}}$ denotes the n-fold application of the fixpoint operator $\mathrm{T}_{\mathcal{P}}$ and $\mathrm{T}^0_{\mathcal{P}} \stackrel{\text{def}}{=} A$. That is, M is the least fixpoint of $\mathrm{T}_{\mathcal{P}}$ applied to A and it expresses all items that are true under the program;we refer to those items as **theorems**. In this paper, we are interested in the task of showing that a given item c'_G is a theorem, i.e., to *prove* c'_G. Fig. 1 gives an illustration of the problem setup in natural language.

2.2 Theorem Proving as A* Search over a Hypergraph

The formalization introduced in the previous section allows us to define a search space over valid deductive inferences, using the structure of a hypergraph (Klein & Manning, 2001). By assigning a cost to inferences, we can define a weight function over paths in the hypergraph, specifically those that lead to (i.e., *prove*) a goal theorem of interest. We can then employ classical search algorithms such as A* search to efficiently find such paths. Our presentation of A* on hypergraphs is subsumed by the generalized framework of Felzenszwalb & McAllester (2007) and is analogous to A* parsers for context-free grammars (Klein & Manning, 2002; 2003).

Hypergraphs and Proofs. We associate each program $\mathcal{P} = \mathcal{R} \sqcup A$ with a **B-hypergraph** (Gallo et al., 1993), which is a (V, E) tuple where V is a set of vertices and $E \subseteq 2^V \times V$ is a set of backwards-hyperedges, each of the form $T \rightarrowtail v_c$, where $T = \{v_{b_1}, ..., v_{b_K}\} \subseteq V$ is called a **tail** and $v_c \in V$ is called a **head**. Additionally, we label each vertex with an element of the minimal Herbrand model M

using a **labeler** $\lambda\colon V \to M$. The vertex labelings must conform to constraints imposed by the axioms and rules of the program: (1) for every vertex v with no incoming hyperedge we require that the label is an axiom, i.e., $\lambda(v) \in A$, and (2) for every hyperedge $e = \{v_{b_1}, ..., v_{b_K}\} \rightarrowtail v_c \in E$, there must exist a rule $(b_1, ..., b_K \vdash c) \in \mathcal{R}$ and a substitution $\theta \in \Theta(\mathcal{P})$ such that $b_1/\theta = \lambda(v_{b_1}), ..., b_K/\theta = \lambda(v_{b_K})$ and $c/\theta = \lambda(v_c)$. That is, search *starts* from the axioms and *proceeds* by traversing the hypergraph through rule applications based on known theorems. To prove a goal theorem c'_G, the search objective is to find a **hyperpath** from a source set $S \subseteq \lambda^{-1}(A)$ (i.e., a subset of the axioms) to a goal vertex $v_G \in V$ labeled with c'_G, which is a sequence of hyperedges $T_1 \rightarrowtail v_{c_1}, ..., T_J \rightarrowtail v_{c_J}$ such that $v_{c_J} = v_G$ and, for every $j \in [J], T_j \subseteq S \cup \{v_{c_1}, ..., v_{c_{j-1}}\}$; we refer to such a hyperpath π as a **proof** of c'_G.

Shortest Proof and Weights. Let $\mathbb{R}_{\geq 0}$ be the nonnegative reals extended with ∞. We associate a **weight** $w\colon M \to \mathbb{R}_{\geq 0}$ with each vertex label, defined recursively as follows: (1) $w(b') \stackrel{\text{def}}{=} 0$ for all axioms $b' \in A$ and (2) $w(c') \stackrel{\text{def}}{=} \min_{b'_1,...,b'_K \vdash c'} g(w(b'_1), ..., w(b'_K))$ for all $c' \in M \setminus A$, where $g\colon \mathbb{R}^K_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a rule-specific **cost function**. The goal of search is to find the weight $w^*_G \stackrel{\text{def}}{=} w(c'_G)$, i.e., the size of the shortest proof of a goal theorem c'_G. There exist many notions of proof size, e.g., depth, number of vertices, or more general rule-specific aggregations; which notion is used is directly determined by the choice of cost functions. In this work, for a rule $b_1, ..., b_K \vdash c$ we set

$$g(w(b'_1), ..., w(b'_K)) \stackrel{\text{def}}{=} 1 + \max_{k=1}^{K} w(b'_k), \tag{2}$$

for all corresponding ground instantiations $b'_1, ..., b'_K \vdash c'$. This yields a notion of size as depth, so the shortest proof will be the one with the shallowest depth (see Knuth, 1977, §2B).[1]

A* Search. A* search is a forward-chaining algorithm where the order in which vertices are expanded is determined by a particular priority queue. Given a program $\mathcal{P} = \mathcal{R} \sqcup A$ and a goal theorem c'_G, it returns the size w^*_G of the shortest proof of c'_G, or ∞ if c'_G cannot be proved. The priority of an item c' depends on both the cost-so-far and the cost-to-go through the following **priority function**:

$$f(c') = w(c') + h(c'), \tag{3}$$

where the **heuristic function** $h\colon M \to \mathbb{R}_{\geq 0}$ estimates the true cost-to-go to the goal from c'.[2] Setting $h(c') = 0$ for all c' corresponds to uninformed search and recovers Dijkstra's (1959)'s algorithm.

We give pseudocode for A* search in Alg. 1. The algorithm maintains a priority queue Q over theorems—called the **agenda**—and a set $\mathcal{C}$ of theorems—called a **chart**. Two operations will be particularly important for the remainder of this paper: *pushes* and *pops*. Each iteration *pops* the item b' from Q with the lowest priority $f(b')$ according to Eq. (3), adds it to the chart, and checks whether new rules can be fired combining b' with the items existing in the chart. Any new conclusions are *pushed* to Q along with their appropriate priorities. The algorithm terminates on popping the goal item.[3]

Guarantees on Correctness and Efficiency. A heuristic function h is **consistent** (or **monotone**) if for all $(b_1, ..., b_K \vdash c) \in \mathcal{R}$ and $\theta \in \Theta(\mathcal{P})$ with $b'_k = b_k/\theta$ and $c' = c/\theta$ such that $b'_1, ..., b'_K, c' \in M$,

$$w(b'_k) + h(b'_k) \leq g(w(b'_1), ..., w(b'_K)) + h(c'), \quad \forall k = 1, ..., K, \tag{4}$$

for all weight assignments $w(b'_1), ..., w(b'_K)$ possible under the algorithm. Intuitively, this means that the priority function f is monotonically nondecreasing along any hyperpath. If h is consistent, it follows that for all items c', the derived weight equals the true weight when c' is added to the chart at line 11 in Alg. 1 (Felzenszwalb & McAllester, 2007, Thm 2). This implies correctness: the weight $w(c'_G)$ returned by Alg. 1 is the weight w^*_G of the shortest proof of c'_G. In addition, given a consistent heuristic function, A* is **optimally efficient** in the sense that it pops the fewest number of items (line 9 in Alg. 1) required to prove the goal by any algorithm with access to the same heuristic information (Dechter & Pearl, 1985). The choice of heuristic function plays a significant role for efficiency in practice, which our experiments explore in the context of LLM reasoning (§4).

[1] We give more details on cost functions and compare our choice to another plausible one in §C.2.

[2] The true cost-to-go is the outside weight $Z(c')$ (Baker, 1979; Lari & Young, 1990). Let $w^*_{G^{c'}}$ be the weight of the shortest proof of c'_G that *passes through* c'. The outside weight for c' is then defined as the value $Z(c')$ such that $w(c') + Z(c') = w^*_{G^{c'}}$, or ∞ if $w^*_{G^{c'}}$ does not exist. See Gildea (2021) for further discussion.

[3] We have omitted backpointers from the pseudocode, which are used to return the shortest proof itself in addition to its size. Our code implementation also returns the trace of logical inferences that were made during search, as will be discussed in §3.1.

Algorithm 1 Generalized A* Search for Deductive Reasoning over Hypergraphs

1. **function** A*($\mathcal{P} = \mathcal{R} \sqcup A, c'_G$)
2. ▷*Program $\mathcal{P} = \mathcal{R} \sqcup A$; goal theorem c'_G*
3. $\mathcal{C} \leftarrow \varnothing; Q \leftarrow \varnothing$ ▷*initialize the chart $\mathcal{C}$ (a set) and the agenda Q (a priority queue)*
4. **for** $b' \in M$: $w(b') \leftarrow \infty$ ▷*default item weights are ∞ (not yet derived)*
5. **for** $b' \in A$: ▷*push axioms to agenda*
6. $w(b') \leftarrow 0$ ▷*axioms have weight w of zero*
7. $Q.\text{PUSH}\big(b',\ h(b')\big)$ ▷*axiom priorities are only based on heuristic values h*
8. **while** $Q \neq \varnothing$:
9. $b' \leftarrow Q.\text{POP}()$ ▷*pop the item with lowest priority (smallest value of $g(\cdot, ..., \cdot) + h(\cdot)$)*
10. **if** $b' \in \mathcal{C}$: **continue** ▷*the item has already been processed*
11. $\mathcal{C} \leftarrow \mathcal{C} \cup \{b'\}$ ▷*add item to chart*
12. **if** $b' = c'_G$: **return** $w(c'_G)$
13. ▷*the algorithm terminates when the goal is popped; correctness ensures that $w(c'_G) = w^*_G$*
14. **for** $(b_1, ..., b_K \vdash c) \in \mathcal{R}$: ▷*check whether b' has unlocked new rule applications*
15. **for** $\theta \in \Theta(\mathcal{P})$: $b_1/\theta, ..., b_K/\theta, c/\theta \in H$ **and** $\forall k \in \{1, ..., K\}$: $b_k/\theta \in \mathcal{C}$: ▷*check premises*
16. $c' \leftarrow c/\theta$ ▷*new conclusion is derived*
17. $w' \leftarrow g(w(b_1/\theta), ..., w(b_K/\theta))$ ▷*compute the weight of new conclusion*
18. **if** $c' \notin \mathcal{C}$ **and** $w' < w(c')$: ▷*new weight is cheaper than known weight*
19. $w(c') \leftarrow w'$ ▷*update the weight*
20. $Q.\text{PUSH}\big(c',\ w' + h(c')\big)$ ▷*push to the agenda*
21. **return** $w(c'_G)$ ▷*goal not provable; the weight is ∞*

3 From Symbolic Search To Language Model Reasoning

The previous section presented background on A* search under a symbolic framework for deductive reasoning. In this section we bridge the gap between reasoning under logic programs and reasoning with natural language tokens as generated by LLMs (§3.1). We then discuss our method for deriving signals from A* search for reward modeling (§3.2).

3.1 Representing Deductive Search in Natural Language

LLMs reason with strings over natural language tokens. We therefore require a way of mapping the execution of the algorithm into a natural language string. We discuss our approach and contrast it to Lehnert et al. (2024), who trained transformers from scratch on A* traces for classical planning tasks.

Tracing the Execution. How can the execution of Alg. 1 be represented as a sequence? Lehnert et al. (2024) propose one answer: write out the items that were popped (line 9 in Alg. 1) and pushed (lines 7 and 20 in Alg. 1), along with corresponding values of w and h (Eq. (3)), following the order in which they occur during execution. However, while they append the proof returned by the algorithm when creating training data, there is generally no way of determining the proof based solely on their notion of execution trace, without additional bookkeeping. In particular, when pushing an item c', it is necessary to know the tail vertices $b'_1, ..., b'_K$ from which c' was derived—so-called **backpointers**.

Our goal is to verify that an LLM's proof is correct, which requires stating *why* each theorem in the proof is true. We therefore propose an alternative to Lehnert et al. (2024) where we represent each push as a **proof step**, which includes the necessary backpointers and the inference rule that was applied. Formally, for a given (program, goal)-pair $(\mathcal{P}, c'_G)$, with substitutions Θ and minimal Herbrand model M (see §2.1 for definitions), consider the following alphabet of symbols:

$$\Delta \stackrel{\text{def}}{=} \{\text{PUSH}((b_1/\theta, ..., b_K/\theta), r, c/\theta) \mid b_1/\theta, ..., b_K/\theta, c/\theta \in M, r = (b_1, ..., b_K \vdash c) \in \mathcal{R}, \theta \in \Theta\}. \tag{5}$$

That is, Δ contains symbols corresponding to all valid proof steps. Let Δ^* be the Kleene closure of Δ, i.e., the set of all strings formed by symbols in Δ. We consider a **search trace** $\boldsymbol{\delta}$ to be an element $\boldsymbol{\delta} \in \Delta^*$ such that the constraints of Alg. 1 are satisfied, i.e., the symbols in $\boldsymbol{\delta}$ correspond to pushes executed by the algorithm in the right order. These traces are readily obtained by additional bookkeeping in Alg. 1.

Pushes and Pops. A search trace as we have defined it does not explicitly contain pops, but it does carry information about them: For a step $\text{PUSH}((b'_1, ..., b'_K), r, c')$, we know that all items $b'_1, ..., b'_K$ must have been popped because Alg. 1 requires that they all be in the chart. We are not concerned with the actual pop ordering of $b'_1, ..., b'_K$ since this ordering does not matter for whether a rule can be applied: as long as all premises are popped before the conclusion, we are agnostic to which ordering the LLM may have popped the premises. We can derive the minimum number of items that must have been popped to yield a given search trace by counting the unique premises that appear in the search trace, i.e., the number of elements $|\mathcal{V}_{\text{pops}}(\boldsymbol{\delta})|$ in the set

$$\mathcal{V}_{\text{pops}}(\boldsymbol{\delta}) \stackrel{\text{def}}{=} \bigcup_{\text{PUSH}((b'_1,...,b'_K),r,c') \in \boldsymbol{\delta}} \{b' \mid b' \in \{b'_1, ..., b'_K\}\}. \tag{6}$$

This is the *minimum* number of required pops since there may have been additional pops that did not result in any push. Empirically, we evaluate efficiency both in terms of proof steps (pushes) and (minimal) pops. Our notion of search trace is an equivalence class on pop orderings that yield the same push ordering. See §C.1 for details and a more formal comparison to Lehnert et al. (2024).

Verbalized Search Traces. Let Γ be an alphabet of natural language tokens. A **language model** p is a probability distribution over the Kleene closure of Γ, denoted Γ^*. To bridge the search trace with natural language we associate each item and rule with a set of strings. For a program $\mathcal{P}$, we define a **verbalizer** $\nu_{\mathcal{P}} \colon H^* \cup \mathcal{R} \to 2^{\Gamma^*}$, where H^* is the Kleene closure of the Herbrand base H and each $\nu_{\mathcal{P}}(b')$ must be a disjoint set. A search trace is verbalized by applying $\nu_{\mathcal{P}}$ to all items and rules in each of the proof steps in the order as they occur, separating the proof steps by a distinguished delimiter sequence—in our case "\n\n". Axiom verbalizations are included in the prompt and are not repeated in the verbalized search trace. See §D.2 for an example on how this is done in our experiments. Fig. 1 gave an informal example of verbalized search traces from A* with the true cost-to-go and Dijkstra's algorithm.

3.2 Deductive Search in Natural Language as a Reinforcement Learning Problem

Our aim is to train LLMs to generate correct and efficient verbalized search traces. This is an instantiation of the learning to search paradigm (Daumé III & Marcu, 2005; Daumé III et al., 2009; Ross et al., 2011), applied to natural language reasoning. We consider both imitation learning through supervised fine-tuning and a reinforcement learning (RL) approach. For the RL approach, the verbalized search trace provides fine-grained signals with regard to whether the proof steps taken were correct and whether the search trace corresponds to the execution of an efficient algorithm. These signals can be quantified through the priority function of A* search from Eq. (3); see Gehring et al. (2022) for related ideas. Note that we have access to ground-truth priority scores at training and inference time by parsing the verbalized search trace and calling Alg. 1. Thus, we do not need to train a neural reward model from data; however, future work may explore using A* search traces to that end.

A*-informed Reward Shaping. Let $(\mathcal{P}, c'_G)$ be a (program, goal)-pair and $\boldsymbol{\delta}$ be a candidate search trace, e.g., corresponding to an LLM solution. We obtain a raw efficiency score by summing the priority values over all items that we know have been popped: $x_{\boldsymbol{\delta}} \stackrel{\text{def}}{=} \sum_{b' \in \mathcal{V}_{\text{pops}}(\boldsymbol{\delta})} f(b')$, where $\mathcal{V}_{\text{pops}}$ is the minimum set of popped items entailed by $\boldsymbol{\delta}$ (§3.1) and f is from Eq. (3). Such raw scores are inappropriate for reward modeling, however, since their range can vary drastically across programs and goals, which could lead to unstable gradient optimization (van Hasselt et al., 2016; Schaul et al., 2021). Moreover, standard RL objectives are built for maximizing rewards, not minimizing costs. In our experiments, we therefore normalize the raw scores with an exponentially decaying reward model:

$$r(\boldsymbol{\delta}) \stackrel{\text{def}}{=} \begin{cases} \exp\left(\beta \cdot \left(\alpha - \sum_{b' \in \mathcal{V}_{\text{pops}}(\boldsymbol{\delta})} f(b')\right)\right), & \text{if } \boldsymbol{\delta} \text{ correctly proves } c'_G \text{ from } \mathcal{P} \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

with scale and location parameters $\beta \in \mathbb{R}_{>0}$ and $\alpha \in \mathbb{R}_{\geq 0}$. The intuition behind Eq. (7) is to give a non-zero reward if the search trace is correct; if it is, the reward is scaled based on efficiency scores as assigned by A*'s priority function. We set α and β adaptively based on $(\mathcal{P}, c'_G)$. Specifically, we set $\alpha = x_{\boldsymbol{\delta}^*}$, where $\boldsymbol{\delta}^*$ is the true search trace returned by A* on input $(\mathcal{P}, c'_G)$. This ensures a reward of 1 if the candidate trace is perfectly aligned with the underlying A* algorithm. Note, however, that the reward will be greater than 1 if the output is more efficient than the underlying A* algorithm,

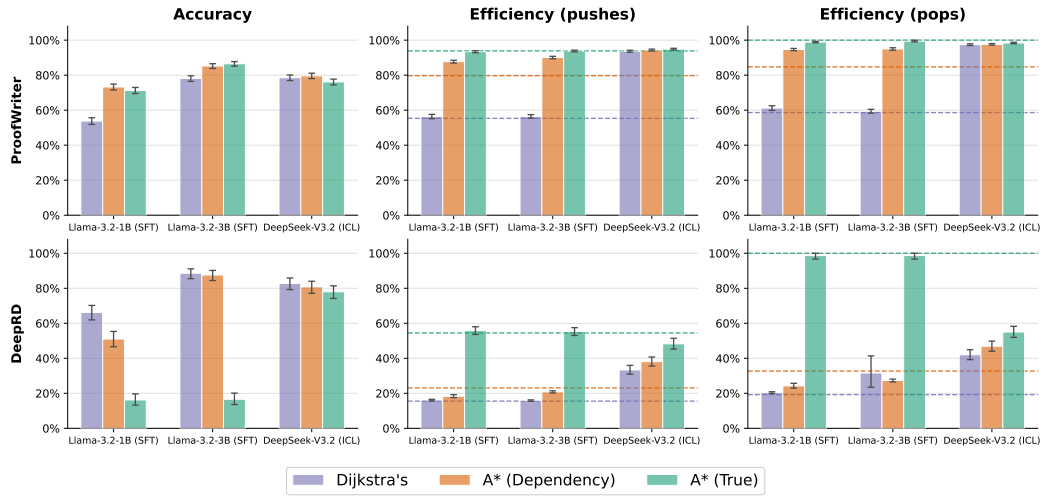

Figure 2: Performance of LLMs finetuned with SFT across different search orders, as compared to an ICL baseline with DeepSeek-V3.2, evaluated on the test sets of ProofWriter and DeepRD. The dashed lines indicate the efficiency scores for the corresponding search algorithm that matches the color. Intervals are 95% Wilson score CIs for accuracy and bootstrap CIs for efficiency.

whose efficiency depends on the heuristic function (discussed below). We set $\beta = \log(2)/x_{\boldsymbol{\delta}^*}$, which results in a scaling behavior in which doubling the score compared to $x_{\boldsymbol{\delta}^*}$ yields half the reward.

Heuristic Functions. The above reward model is compatible with arbitrary A* heuristics, the choice of which is important for the practical efficiency of A* (Russell & Norvig, 2021). A heuristic that is closer to the true cost-to-go will generally lead to more efficient search in terms of number of pops. A common way to derive heuristics is to compute the cost-to-go for relaxed variants of the problem, in which fewer restrictions are enforced on when certain rules are applicable. In other words, a new hypergraph (V', E') is constructed for which the original hypergraph (V, E) is a subhypergraph, i.e., $V \subseteq V'$ and $E \subseteq E'$, where the cost functions associated with hyperedges in (V, E) are preserved in (V', E'). The cost-to-go from the relaxed problem will be a consistent heuristic (Felzenszwalb & McAllester, 2007). Following this idea, we adopt a heuristic based on the unary dependencies encoded in the program's rules. For all ground instantiations $b'_1, ..., b'_K \vdash c'$, we add K edges $\{v_{b_1}\} \rightarrowtail v_c, ..., \{v_{b_K}\} \rightarrowtail v_c$, where $v_{b_1}, ..., v_{b_K}, v_c$ are the corresponding vertices in the original hypergraph. Intuitively, this heuristic encodes the dependency of the rules without considering which theorems are entailed by the axioms; we call it the **dependency heuristic**. Our experiments additionally use the true cost-to-go and the zero heuristic (Dijkstra's algorithm) as heuristics.

4 Can A* Training Yield Improvements in Reasoning?

This section addresses the paper's main empirical question. §4.1 explains the experimental setup and §4.2-4.3 discuss the results.

4.1 Experimental Setup

Datasets. We take logic programs and goal theorems from ProofWriter (Tafjord et al., 2021) and DeepRD (Rameshkumar et al., 2025). For ProofWriter, we only consider problems where the depth of the shortest proof is at least 3 and the goal is provable under the program, resulting in 9207 training examples, 1352 validation examples, and 2615 test examples. DeepRD is a programmable dataset which supports arbitrarily complex programs along two dimensions, a lookahead value L determining the depth of the shortest proof and a branching factor B determining the number of branches from the single axiom vertex. We generate 2000 training examples, 200 validation examples, and 500 test examples using values $L \in \{5, ..., 10\}$ and $B \in \{4, ..., 8\}$. Figs. 5 and 6 show the number of proof steps (i.e., pushes) performed by the three different search orders on ProofWriter and DeepRD, respectively. We make two observations: (1) the DeepRD problems are considerably longer than the ProofWriter problems and (2) the dependency heuristic performs similarly to the true cost-to-go on ProofWriter, whereas on DeepRD the dependency heuristic gives little advantage over Dijkstra's.

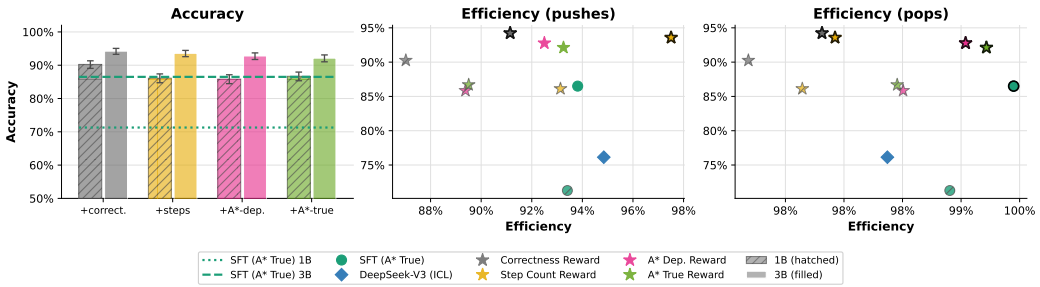

Figure 3: Results on **ProofWriter** for GRPO-finetuned models, starting from SFT checkpoints that were trained on A* traces under the *true cost-to-go*. Points on the Pareto front are highlighted. We observe that all reward models are placed on at least one of the two accuracy-efficiency Pareto fronts.

Evaluation. Importantly, we evaluate the entire verbalized search trace generated by the LLM. In contrast, most prior work treats natural language theorem proving as a simplified task such as binary classification (e.g., Clark et al., 2021). We parse the LLM's output into search traces using a specified format for segmentation; §D.2 gives an example. We allow arbitrary token sequences between the blocks of verbalized proof steps and consider variations of verbalizations beyond what is in the training data. Solutions are scored on correctness and efficiency. Correctness (**accuracy**) is a binary score; a search trace is considered correct if it yields a valid proof of the goal under the axioms and rules of the program. That is, any incorrect proof step required to prove the goal will result in a score of 0. Efficiency is scored by comparing *correct* proofs against the ground-truth shortest (i.e., shallowest) proof of the problem, following Opedal et al. (2025). We consider two variants: **efficiency (pushes)**, which measures the number of proof steps (or pushes), and **efficiency (pops)**, which measures the minimum number of popped items required to yield the search trace. For both metrics we take the proportion against the corresponding number in the shortest proof. Note that the accuracy and efficiency scores naturally trade off: efficiency can be optimized by only being correct on easy proofs, while accuracy can be optimized by ignoring efficiency as a target metric.

Search Orders and Training. We finetune Llama-3.2-1B-Instruct and Llama-3.2-3B-Instruct (Llama Team, 2024) using supervised fine-tuning (SFT) and RL with GRPO (Shao et al., 2024) and the reward model introduced in §3.2. As baselines we consider in-context learning (ICL) of the base models and of DeepSeek-V3.2 (Liu et al., 2025). To obtain data, we feed the (program, goal)-pairs to our implementation of Alg. 1 and translate the resulting search traces into verbalizations using templates taken from the datasets. We verbalize search traces according to three different search heuristics: (i) $h = 0$ (Dijkstra's uninformed search), (ii) A* with the dependency heuristic, and (iii) A* with the true cost-to-go oracle, which yields the same number of pops as the shortest proof and is therefore close to a solution-only baseline.[4] For RL we consider four different reward models: (i) a **correctness** reward that assigns 1 if the verbalized search trace correctly proves the goal and 0 otherwise, (ii) a **step count** reward which instantiates Eq. (7) as an exponential decay over the number of correct verbalized proof steps, where α is set as the number of steps in the shortest proof, and A*-informed reward models which instantiate Eq. (7) with (iii) the dependency heuristic and (iv) the true cost-to-go oracle. Additional details such as hyperparameter values are given in §D.3.

4.2 Imitation of Search with Supervised Fine-tuning

We first note that the Llama-3.2-1B and Llama-3.2-3B models are unable to perform the tasks without training—the ICL baseline accuracies are $\approx 0\%$, so we omit them from the figures. Anecdotally, we observe that many of their responses contain hallucinated rules and inferences. The results for SFT are presented in Fig. 2. When trained on the ProofWriter data, the 3B model outperforms the DeepSeek baseline with A* fine-tuning, whereas the 1B model shows somewhat lower accuracy. Using a heuris-

[4] In practice, several items often have the same priority value. We can ensure that the true cost-to-go traces are the shortest proofs through the following tie-breaking strategy: firstly by value of heuristic estimate (lower is better) and secondly by last-in-first-out (Asai & Fukunaga, 2017). Since we are in the uniform cost setting, in which all hyperedges are weighted equally, such "depth-first" tie-breaking yields the fewest number of pops among all tie-breaking continuations; see Fukunaga (2026, Thm. 2). This implies that the search trace under the true cost-to-go is the shortest proof in terms of number of pops.

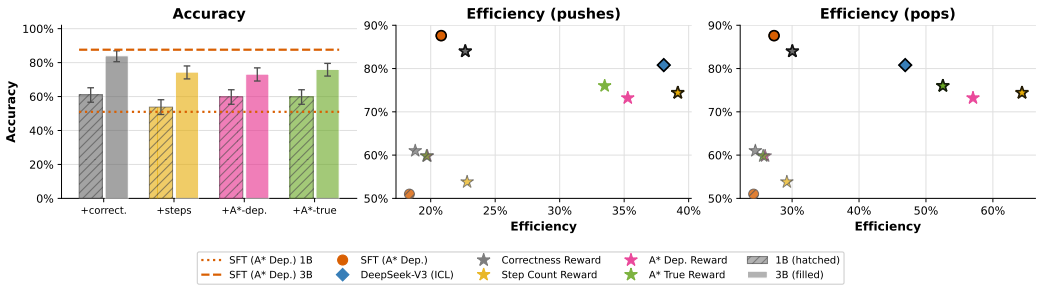

Figure 4: Results on **DeepRD** for GRPO-finetuned models, starting from SFT checkpoints that were trained on A* traces under the *dependency heuristic*. Points on the Pareto front are highlighted. The accuracy-efficiency trade-off is shown even more starkly than in Fig. 3.

tic outperforms Dijkstra's algorithm in terms of both efficiency and correctness. There is no significant difference in accuracy between the dependency heuristic and the true cost-to-go oracle on this dataset.

For DeepRD, which has a considerably larger search space than ProofWriter (see Figs. 5 and 6), SFT exposes a trade-off between accuracy and efficiency. The true cost-to-go does poorly on accuracy but the problems that are solved are done so efficiently. Dijkstra's, on the other hand, generalizes best to the test set in terms of accuracy—with the 3B model beating DeepSeek-V3.2—but is restricted by inefficient search. We emphasize, however, that efficiency can only be evaluated on examples where the proof is correct. These results suggest that learning the true heuristic is hard if the search space is large. This result is consistent with and extends previous findings that learning to search outperforms training on optimal paths (Gandhi et al., 2024; Lehnert et al., 2024).

4.3 Reinforcement Learning for Efficient Search

Fig. 3 shows performance on ProofWriter data resulting from GRPO fine-tuning from the true cost-to-go SFT checkpoint with the various reward models. All reward models lead to improved accuracy over the corresponding SFT baseline, with the 1B models showing accuracy comparable to or even better than the SFT-tuned 3B model. The correctness reward yields the best correctness performance overall. As expected, however, the other three rewards yield better efficiency than the simple correctness reward. When measuring efficiency in terms of pops—which A* search optimizes for—the two A* rewards are preferable; they achieve the highest efficiency and are both on the Pareto front when comparing to other models of the same size. When measuring push efficiency, however, the step count reward is preferable. In general, no reward model dominates any of the others on both accuracy and efficiency.

Fig. 4 shows analogous results on the DeepRD dataset, with the difference that the SFT checkpoint is based on A* with the dependency heuristic rather than the true cost-to-go. In general, the two A* rewards strike a balance between correctness and efficiency: the step count reward yields efficient proofs but mostly lower accuracy than A* rewards, while the correctness reward yields more accurate but less efficient proofs. Interestingly, while all rewards dominate the SFT checkpoint for the 1B model, the 3B SFT model appears on the Pareto front as the most accurate option. We also note that, while the models are less competitive against DeepSeek-V3.2 on this data, DeepSeek-V3.2 does not dominate. Figs. 8 and 9 in §E show further results.

5 Conclusion

Our paper framed deductive reasoning in natural language as a task where the entire proof must be correct. Building on the learning-to-search paradigm, we explored whether signals from A* search can be used to train models to prove theorems correctly and efficiently. We considered several variations of search heuristics, reward models, and datasets. Overall, we discovered a good recipe in starting with supervised fine-tuning on A* traces under a useful but imperfect heuristic, and continuing with reinforcement learning using a reward model informed by correctness and/or efficiency—the choice of which should depend on what is prioritized. Our framework, analysis, and results advance our understanding about training LLMs to perform search, demonstrating potential for methods guided by classical planning algorithms.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology and experiments, implement methods, clean and reformat datasets, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, or assist with translation. Additionally, we used generative AI tools to create or modify scientific figures or images, suggest experimental parameters, create or edit software code, draft parts of a research paper, discover research topics or identify gaps, and brainstorm. We have reviewed all AI-assisted work. For example, we verified and tested all LLM-generated code and all LLM-generated text has been thoroughly edited. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=L8094Whth0.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Proceedings of the ICML Workshop ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=tCUGSPSZ3A.

Masataro Asai and Alex Fukunaga. Tie-breaking strategies for cost-optimal best first search. *Journal of Artificial Intelligence Research*, 58(1):67–121, January 2017. ISSN 1076-9757. doi: 10.1613/jair.5249. URL https://doi.org/10.1613/jair.5249.

Alex Ayoub, Kavosh Asadi, Dale Schuurmans, Csaba Szepesvari, and Karim Bouyarmane. Learning to reason efficiently with discounted reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=R4GPttoDyn.

James K. Baker. Trainable grammars for speech recognition. In *Speech Communication Papers for the Meeting of the Acoustical Society of America*, 1979. URL https://compcalc.github.io/public/baker/baker1979trainable.pdf.

Leonardo Bertolazzi, Albert Gatt, and Raffaella Bernardi. A systematic analysis of large language models as soft reasoners: The case of syllogistic inferences. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13882–13905, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.769. URL https://aclanthology.org/2024.emnlp-main.769/.

Leonardo Bertolazzi, Sandro Pezzelle, and Raffaella Bernardi. How language models conflate logical validity with plausibility: A representational analysis of content effects. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9316–9347, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.454. URL https://aclanthology.org/2026.findings-acl.454/.

Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, March 1989. URL https://doi.org/10.1109/69.43410.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? On the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=MSbU3L7V00.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165. URL https://dl.acm.org/doi/abs/10.5555/3491440.3491977.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3Pf3Wg6o-A4.

Hal Daumé III and Daniel Marcu. Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 169–176, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102373. URL https://doi.org/10.1145/1102351.1102373.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, Jun 2009. ISSN 1573-0565. doi: 10.1007/s10994-009-5106-x. URL https://doi.org/10.1007/s10994-009-5106-x.

Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 32(3):505–536, July 1985. ISSN 0004-5411. doi: 10.1145/3828.3830. URL https://doi.org/10.1145/3828.3830.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 (1):269–271, 1959. URL https://doi.org/10.1007/BF01386390.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang (Lorraine) Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/deb3c28192f979302c157cb653c15e90-Paper-Conference.pdf.

Tiwalayo Eisape, Michael Tessler, Ishita Dasgupta, Fei Sha, Sjoerd van Steenkiste, and Tal Linzen. A systematic comparison of syllogistic reasoning in humans and language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8425–8444, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.466. URL https://aclanthology.org/2024.naacl-long.466/.

Jason Eisner. Time-and-space-efficient weighted deduction. *Transactions of the Association for Computational Linguistics*, 11:960–973, 2023. doi: 10.1162/tacl_a_00588. URL https://aclanthology.org/2023.tacl-1.54/.

Pedro F. Felzenszwalb and David McAllester. The generalized A* architecture. *Journal of Artificial Intelligence Research*, 29(1):153–190, June 2007. ISSN 1076-9757. URL https://dl.acm.org/doi/abs/10.5555/1622606.1622612.

Alex Fukunaga. Expansion counts under standard A* tie-breaking strategies on the final plateau, 2026. URL https://arxiv.org/abs/2609.23293.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177–201, 1993. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(93)90045-P. URL https://www.sciencedirect.com/science/article/pii/0166218X9390045P.

Kanishk Gandhi, Denise H. J. Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. Stream of search (SoS): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Kanishk Gandhi, Ayush K. Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Clement Gehring, Masataro Asai, Rohan Chitnis, Tom Silver, Leslie Kaelbling, Shirin Sohrabi, and Michael Katz. Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. In *Proceedings of the international conference on automated planning and scheduling*, volume 32, pp. 588–596, 2022. URL https://ojs.aaai.org/index.php/ICAPS/article/view/19846/19605.

Daniel Gildea. Efficient outside computation. *Computational Linguistics*, 46(4):745–762, 02 2021. ISSN 0891-2017. doi: 10.1162/coli_a_00386. URL https://doi.org/10.1162/coli_a_00386.

Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999. URL https://aclanthology.org/J99-4004.

Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, R. Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, A. Liu, Bing Xue, Bing-Li Wang, Bochao Wu, B. Feng, Chengda Lu, Chenggang Zhao, C. Deng, Chenyu Zhang, C. Ruan, Damai Dai, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, JingChang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. Cai, J. Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, K. Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, M. Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. Li, Shuang Zhou, Shao-Kang Wu, Tao Yun, Tian Pei, T. Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, W. Liang, Wenjun Gao, Wen-Xia Yu, Wentao Zhang, W. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, X. Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xi-Cheng Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Y. Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Y. Ou, Yuduan Wang, Yue Gong, Yu-Jing Zou, Yujia He, Yunfan Xiong, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yao Li, Yi Zheng, Yuchen Zhu, Yunxiang Ma, Ying Tang, Y. Zha, Yuting Yan, Z. Ren, Z. Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Devaansh Gupta and Boyang Li. A training data recipe to accelerate A* search with language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6681–6695, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.391. URL https://aclanthology.org/2024.findings-emnlp.391/.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, Miami,

Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1229. URL https://aclanthology.org/2024.emnlp-main.1229/.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136. URL https://doi.org/10.1109/TSSC.1968.300136.

John M. Howie. *Fundamentals of Semigroup Theory*. Oxford University Press, 12 1995. ISBN 9780198511946. doi: 10.1093/oso/9780198511946.001.0001. URL https://doi.org/10.1093/oso/9780198511946.001.0001.

Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In Liang Huang (ed.), *Coling 2008: Advanced Dynamic Programming in Computational Linguistics: Theory, Algorithms and Applications - Tutorial notes*, pp. 1–18, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL https://aclanthology.org/C08-5001/.

Neil Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1):86–104, 1986. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(86)80029-8. URL https://www.sciencedirect.com/science/article/pii/S0019995886800298.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4722–4756, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.272. URL https://aclanthology.org/2024.emnlp-main.272/.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. LAMBADA: Backward chaining for automated reasoning in natural language. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6547–6568, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.361. URL https://aclanthology.org/2023.acl-long.361/.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2015. URL http://arxiv.org/abs/1412.6980.

Dan Klein and Christopher D. Manning. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pp. 123–134, Beijing, China, October 2001. URL https://aclanthology.org/W01-1812/.

Dan Klein and Christopher D Manning. Fast exact inference with a factored model for natural language parsing. In S. Becker, S. Thrun, and K. Obermayer (eds.), *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/6c97cd07663b099253bc569fe8d342bb-Paper.pdf.

Dan Klein and Christopher D. Manning. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 119–126, 2003. URL https://aclanthology.org/N03-1016/.

Donald E. Knuth. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5, 1977. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(77)90002-3. URL https://www.sciencedirect.com/science/article/pii/0020019077900023.

Yeganeh Kordi, Nihal V. Nayak, Max Zuo, Ilana Nguyen, and Stephen Bach. Revisiting generalization across difficulty levels: It's not so easy. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7014–7042, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.330. URL https://aclanthology.org/2026.eacl-long.330/.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Hannah R. Sheahan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. 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.

Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4(1):35–56, 1990. ISSN 0885-2308. doi: https://doi.org/10.1016/0885-2308(90)90022-X. URL https://www.sciencedirect.com/science/article/pii/088523089090022X.

Alex Lawsen. Comment on the illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL https://arxiv.org/abs/2506.09250.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul McVay, Michael Rabbat, and Yuandong Tian. Beyond A*: Better planning with transformers via search dynamics bootstrapping. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=SGoVIC0u0f.

Belinda Z. Li, Been Kim, and Zi Wang. Questbench: Can LLMs ask the right question to acquire information in reasoning tasks? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=gpwA9aZLTZ.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165. URL https://www.ijcai.org/proceedings/2020/0501.pdf.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hj9eKpqxQl.

Llama Team. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 780–799, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.57. URL https://aclanthology.org/2022.naacl-main.57/.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL https://openreview.net/forum?id=ioYybCRcyW.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. Towards robust mathematical reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35418–35442, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1794. URL https://aclanthology.org/2025.emnlp-main.1794/.

Lovish Madaan, David Esiobu, Pontus Stenetorp, Barbara Plank, and Dieuwke Hupkes. Lost in inference: Rediscovering the role of natural language inference for large language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9229–9242, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.466. URL https://aclanthology.org/2025.naacl-long.466/.

Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant G Honavar. Brains vs. bytes: Evaluating LLM proficiency in olympiad mathematics. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=uXR2KsA4L9.

Clara Meister, Tim Vieira, and Ryan Cotterell. Best-first beam search. *Transactions of the Association for Computational Linguistics*, 8:795–809, 2020. doi: 10.1162/tacl_a_00346. URL https://aclanthology.org/2020.tacl-1.51/.

Silin Meng, Yiwei Wang, Cheng-Fu Yang, Nanyun Peng, and Kai-Wei Chang. LLM-a*: Large language model enhanced incremental heuristic search on path planning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1087–1102, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.60. URL https://aclanthology.org/2024.findings-emnlp.60/.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in

large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. Learning deductive reasoning from synthetic corpus based on formal logic. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25254–25274. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/morishita23a.html.

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

Andreas Opedal, Yanick Zengaffinen, Haruki Shirakami, Clemente Pasti, Mrinmaya Sachan, Abulhair Saparov, Ryan Cotterell, and Bernhard Schölkopf. Are language models efficient reasoners? A perspective from logic programming. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2510.25626.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

Adam Pauls and Dan Klein. K-best A* parsing. In Keh-Yih Su, Jian Su, Janyce Wiebe, and Haizhou Li (eds.), *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 958–966, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/P09-1108/.

Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunovic, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating LLMs on 2025 USA math olympiad. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL https://openreview.net/forum?id=3v650rMO5U.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. ThoughtTerminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=oHR862dpMC.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2933–2958, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.172. URL https://aclanthology.org/2024.emnlp-main.172/.

Revanth Rameshkumar, Jimson Huang, Yunxin Sun, Fei Xia, and Abulhair Saparov. Reasoning models reason well, until they don't. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 936–956, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.52. URL https://aclanthology.org/2025.ijcnlp-long.52/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Parker Riley and Daniel Gildea. Outside computation with superior functions. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2936–2940, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.233. URL https://aclanthology.org/2021.naacl-main.233/.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2021. URL https://aima.cs.berkeley.edu/.

Abulhair Saparov and He He. 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.

Abulhair Saparov, Srushti Ajay Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Mehran Kazemi, Najoung Kim, and He He. Transformers struggle to learn to search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9cQB1Hwrtw.

Tom Schaul, Georg Ostrovski, Iurii Kemaev, and Diana Borsa. Return-based scaling: Yet another normalisation trick for deep RL, 2021. URL https://arxiv.org/abs/2105.05347.

Bilgehan Sel, Ahmad Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/sel24a.html.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=i5PoejmWoC.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDL planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=1QMMUB4zfl.

Leon Sterling and Ehud Shapiro. *The Art of Prolog: Advanced programming techniques*. MIT Press, Cambridge, MA, USA, 2nd edition, 1994. ISBN 0262193388. URL https://cliplab.org/~logalg/doc/The_Art_of_Prolog.pdf.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (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/.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=bzs4uPLXvi.

Hado van Hasselt, Arthur Guez, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/5227b6aaf294f5f027273aebf16015f2-Paper.pdf.

Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pp. 137–146. Association for Computing Machinery, 1982. ISBN 0897910702. doi: 10.1145/800070.802186. URL https://doi.org/10.1145/800070.802186.

David Wan, Mengwen Liu, Kathleen McKeown, Markus Dreyer, and Mohit Bansal. Faithfulness-aware decoding strategies for abstractive summarization. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2864–2880, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.210. URL https://aclanthology.org/2023.eacl-main.210/.

Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen-tse Huang, Pinjia He, Wenxiang Jiao, and Michael Lyu. LogicAsker: Evaluating and improving the logical reasoning ability of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2124–2155, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.128. URL https://aclanthology.org/2024.emnlp-main.128/.

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

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, Louis Castricato, Jan-Philipp Franken, Nick Haber, and Chelsea Finn. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought, 2025. URL https://arxiv.org/abs/2501.04682.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. Logicreward: Incentivizing LLM reasoning via step-wise logical supervision. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IRhYVOKFe0.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. On the paradox of learning to reason from data. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 3365–3373. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/375. URL https://doi.org/10.24963/ijcai.2023/375. Main Track.

A LIMITATIONS

We emphasize that the scope of this paper is restricted to the subset of deductive reasoning captured by logic programs. Our methods are therefore not applicable to more general deductive reasoning that

cannot be solved in polynomial time (Vardi, 1982; Immerman, 1986), extensions such as functions and negations, as well as other important forms of reasoning such as abductive and inductive reasoning. Future work should study algorithms for solving problems that require multiple forms of reasoning in combination. Furthermore, our evaluation is restricted to in-distribution data. Future studies should investigate whether A*-informed training can yield gains in out-of-distribution as well as out-of-domain generalization. We also note that efficiency can only be computed on correct proofs and is therefore confounded with difficulty of the problem.

Empirically, we are restricted to training models in the 1B-3B range due to limited access to computational resources. For similar reasons, we only perform a single training run per configuration. We are also unable to perform large-scale experiments on expensive, frontier models due to the high costs of running inference on such models.

B Related Work

Logical Reasoning. Many studies have focused on measuring and improving transformer-based LLMs on deductive reasoning tasks in natural language (Liu et al., 2021; Creswell et al., 2023; Kazemi et al., 2023; Han et al., 2024; Wan et al., 2024; Li et al., 2025; Madaan et al., 2025) or comparing their performance to humans (Eisape et al., 2024; Lampinen et al., 2024; Bertolazzi et al., 2024; 2026), with some existing datasets being based on logic programs (Clark et al., 2021; Tafjord et al., 2021; Zhang et al., 2023). However, most of these papers reduce reasoning to a simplified setting, such as binary classification of whether a statement is entailed by a program, multiple-choice question answering, or predicting the next step given a partial verbalized proof. Our introduction argued that LLMs should be evaluated based on the entire proofs; see, e.g., Morishita et al. (2023) and Saparov & He (2023) for previous work in this direction. Several papers have proposed methods to improve reasoning by incorporating symbolic solvers that are external to the LLM (Olausson et al., 2023; Pan et al., 2023; Quan et al., 2024). The approach by Xu et al. (2026) is related to our work; they use a reward model based on a theorem prover for scoring logical validity. However, they evaluate performance in terms of the accuracy of the final answer.

The "Overthinking" Problem. LLMs post-trained to solve reasoning tasks with RLVR (so-called reasoning models; Guo et al., 2025) have been found to be subject to "overthinking"—that is, generating solution traces that are far longer than what should intuitively be required to solve the problem (Chen et al., 2025; Pu et al., 2025). Several papers have aimed to address this by incorporating rewards based on the length of the generated token sequence (Arora & Zanette, 2025; Luo et al., 2025; Ayoub et al., 2026; Liu et al., 2026, *inter alia*). This issue is related, but separate from the notion of efficiency considered in this paper: An LLM trained to find the correct final answer with few tokens may still perform incorrect proof steps or take shortcuts. Conversely, an LLM trained to find an efficient proof, e.g., using our method, may still require a long sequence of reasoning tokens to do so.

Search and the A* Algorithm. Several recent papers have advocated a learning-to-search approach to LLM reasoning (Gandhi et al., 2024; Sel et al., 2024; Shah et al., 2024; Saparov et al., 2025; Xiang et al., 2025). We argue that it is desirable to constrain search to inferences that are *correct*. Our method is most related to Lehnert et al. (2024), who found that training transformers from scratch on A* search traces outperforms solution-only training on synthetic planning tasks. We explore whether A*-based training leads to gains in a different setting: fine-tuning larger models on deductive reasoning in natural language, using both imitation learning via SFT as well as RL via A*-informed rewards. We are unaware of previous work that trains LLMs on A* search traces for deductive reasoning in natural language, which requires an extension of A* to hypergraphs as presented in §2.2. Such A* extensions have previously been applied to context-free parsing (Klein & Manning, 2003; Pauls & Klein, 2009). Other lines of work have employed A* for natural language decoding (Meister et al., 2020; Lu et al., 2022; Wan et al., 2023) and used LLMs in combination with A* to improve search for classical planning tasks (Silver et al., 2022; Gupta & Li, 2024; Meng et al., 2024).

C More Details on A* Search and Verbalizations

We provide more details on A* search as applied here to hypergraphs and verbalized natural language reasoning. §C.1 provides a formal discussion of the verbalized search traces and compares our method to Lehnert et al.'s (2024). §C.2 discusses the choice of cost function.

C.1 Formal Comparison to Lehnert et al. (2024) and Equivalence over Pop Orderings

§3.1 mentioned how there is an equivalence class over orderings of pops resulting from execution of Alg. 1 that lead to the same ordering of pushes. This section formalizes that claim by spelling out how search traces are quotient sets induced by string homomorphisms. We compare the homomorphism considered here to the one corresponding to Lehnert et al.'s (2024) approach.

We fix a program $\mathcal{P}$ and a goal theorem c'_G, with substitutions Θ and minimal Herbrand model M; we refer back to §2.1 for definitions. Consider the following alphabets:

$$\Delta_{\text{POP}} \stackrel{\text{def}}{=} \{\text{POP}(c/\theta) \mid c/\theta \in M\}, \tag{8}$$

$$\Delta_{\text{PUSH}} \stackrel{\text{def}}{=} \{\text{PUSH}(c/\theta) \mid c/\theta \in M\}, \tag{9}$$

$$\Delta_{\text{STEP}} \stackrel{\text{def}}{=} \{\text{PUSH}((b_1/\theta, ..., b_K/\theta), r, c/\theta) \,|\, b_1/\theta, ..., b_K/\theta, c/\theta \in M, r = (b_1, ..., b_K \vdash c) \in \mathcal{R}, \theta \in \Theta\}, \tag{10}$$

That is, Δ_{POP} contains symbols corresponding to pops, while Δ_{PUSH} and Δ_{STEP} contain symbols corresponding to pushes. Δ_{PUSH} and Δ_{STEP} represent pushes in different ways: the former only includes the item that was pushed, whereas the latter additionally includes the premises and the rule that were used to trigger the push. Furthermore, define $\Delta_1 \stackrel{\text{def}}{=} \Delta_{\text{POP}} \sqcup \Delta_{\text{STEP}}$ and $\Delta_2 \stackrel{\text{def}}{=} \Delta_{\text{POP}} \sqcup \Delta_{\text{PUSH}}$.

We consider elements of Δ_1^* and Δ_2^*, i.e., the Kleene closures over Δ_1 and Δ_2, respectively. Further note that the sets Δ_1^* and Δ_2^* form free monoids on Δ_1 and Δ_2, respectively. Let $L_1 \subseteq \Delta^*$ be the set of valid traces, i.e., traces that conform to the constraints of Alg. 1, when representing pushes as proof steps. We use $\boldsymbol{\delta} = \delta_1 \circ ... \circ \delta_{|\boldsymbol{\delta}|}$ to denote elements of L_1, where $\circ$ is the concatenation operator.

We now explain how verbalizing execution traces in L_1 can be defined as a quotient set induced by a string homomorphism, contrasting Lehnert et al.'s (2024) approach to ours. Let us first discuss Lehnert et al. (2024). Their approach corresponds to the homomorphism $\Phi_{\text{L}} \colon L_1 \to \Delta_2^*$ where $\Phi_{\text{L}}(\boldsymbol{\delta}) = \Phi_{\text{L}}(\delta_1) \circ \cdots \circ \Phi_{\text{L}}(\delta_{|\boldsymbol{\delta}|})$ and

$$\Phi_{\text{L}}(\delta) \stackrel{\text{def}}{=} \begin{cases} \delta, & \text{if } \delta \in \Delta_{\text{POP}} \\ \text{PUSH}(c/\theta), & \text{if } \delta = \text{PUSH}((b_1/\theta, ..., b_K/\theta), r, c/\theta) \in \Delta_{\text{STEP}}. \end{cases} \tag{11}$$

That is, Φ_{L} deletes the backpointers that store information about premises and rules from the trace.[5] Note that Φ_{L} is a homomorphism since it preserves the structure of concatenation, i.e., $\Phi_{\text{L}}(\boldsymbol{\delta}_1 \circ \boldsymbol{\delta}_2) = \Phi_{\text{L}}(\boldsymbol{\delta}_1) \circ \Phi_{\text{L}}(\boldsymbol{\delta}_2)$. The homomorphism induces a congruence $\sim_{\text{L}}$ as follows: $\boldsymbol{\delta}_1 \sim_{\text{L}} \boldsymbol{\delta}_2 \Leftrightarrow \Phi_{\text{L}}(\boldsymbol{\delta}_1) = \Phi_{\text{L}}(\boldsymbol{\delta}_2)$ (Howie, 1995). Formally, the set of what is verbalized is the quotient set $L_1/\sim_{\text{L}}$, which is the trace of the algorithm after deleting premises and rules. We remark that using this quotient would be inadequate for the purposes of our paper since it does not recover the proof.

Now, the approach taken here corresponds to the following homomorphism: $\Phi \colon L_1 \to \Delta_1^*$, where $\Phi(\boldsymbol{\delta}) = \Phi(\delta_1) \circ \cdots \circ \Phi(\delta_{|\boldsymbol{\delta}|})$ and

$$\Phi(\delta) \stackrel{\text{def}}{=} \begin{cases} \varepsilon, & \text{if } \delta \in \Delta_{\text{POP}} \\ \delta, & \text{if } \delta \in \Delta_{\text{STEP}}. \end{cases} \tag{12}$$

That is, Φ maps a trace sequence to the corresponding subsequence in which all pops have been deleted. With the congruence $\boldsymbol{\delta}_1 \sim \boldsymbol{\delta}_2 \Leftrightarrow \Phi(\boldsymbol{\delta}_1) = \Phi(\boldsymbol{\delta}_2)$, we verbalize the quotient set $L_1/\sim$, which recovers the trace of the algorithm up to equivalence of pop orderings. We preserve the backpointers in order to be able to construct proofs and evaluate the correctness of the trace. From the backpointers, we can construct the minimal set of popped items required to yield the trace, as discussed in §3.1.

[5] We note that Lehnert et al. (2024) additionally include the values of w and h, which we omit here for brevity.

C.2 On the Choice of Cost Function

Superior Functions. The cost function g can be chosen freely as long as it is **superior** (Knuth, 1977). That is, g must have the following two properties:

$$\text{(i):}\quad g(w(b'_1), ..., w(b'_k), ..., w(b'_K)) \geq g(w(b'_1), ..., w(\hat{b}_k), ..., w(b'_K) \text{ if } w(b'_k) \geq w(\hat{b}_k) \tag{13}$$

$$\text{(ii):}\quad g(w(b'_1), ..., w(b'_K)) \geq \max_{k=1}^{K} w(b'_k). \tag{14}$$

We remark that a common alternative to Knuth's (1977) superior functions framework is to define weights using semirings; see, e.g., Goodman (1999), Huang (2008), and Eisner (2023). However, not all superior functions can be defined with semirings (and vice versa; Gildea, 2021; Riley & Gildea, 2021). Indeed, our particular choice of g as $g(w(b'_1), ..., w(b'_K)) \stackrel{\text{def}}{=} 1 + \max_{k=1}^{K} w(b'_k)$, is one example. As mentioned in §2.2, this yields a notion of shortest proof as the one with the shallowest depth. An alternative, reasonable choice is $g_2(w(b'_1), ..., w(b'_K)) \stackrel{\text{def}}{=} 1 + \sum_{k=1}^{K} w(b'_k)$, which yields a notion of shortest proof as the one with the fewest vertices—this cost function is well-defined under the semiring framework. Note that both of these cost functions reduce to the shortest path for standard weighted graphs.

Shallowest Proof vs. Proof with Fewest Vertices. In this section, we compare the cost function

$$g_1(w(b'_1), ..., w(b'_K)) \stackrel{\text{def}}{=} 1 + \max_{k=1}^{K} w(b'_k) \tag{15}$$

and the alternative

$$g_2(w(b'_1), ..., w(b'_K)) \stackrel{\text{def}}{=} 1 + \sum_{k=1}^{K} w(b'_k) \tag{16}$$

in how they affect the execution traces returned by Alg. 1 and our verbalizations discussed in §3.1 and §C.1. Additionally, in order for Eq. (16) to correspond to the number of vertices we require the weight of the axioms to be 1, i.e., $w(b') = 1$ for all $b' \in A$, rather than 0 as they are for Eq. (15) (§2.2). We measure verbalization length by summing the number of times an item has been verbalized in the trace over all pushes/proof steps. First note that with our notion of execution trace, i.e., with the homomorphism Φ from §C.1 over the alphabet $\Delta_1 \stackrel{\text{def}}{=} \Delta_{\text{POP}} \sqcup \Delta_{\text{STEP}}$ (see Eq. (10)), a pushed item c' with K premises will have a verbalization length of $2K + 2$: K items as premises, $K + 1$ items in the rule, and one conclusion.

First, consider a goal c'_G which has two proofs. One of them uses a single inference rule in one step over five axioms, and the other one uses a linear chain of four inference rules, each with a single premise and starting from a single axiom. The first proof is shortest according to g_1 (the first proof has cost 1 and the second has cost 4), whereas the second proof is the shortest according to g_2 (the first proof has cost 6 and the second has cost 5). The first proof corresponds to the trace

$$\boldsymbol{\delta}_1 = \text{PUSH}((a'_1, ..., a'_5), a_1, ..., a_5 \vdash c_G, c'_G), \tag{17}$$

which has a verbalization length of 12 (we omit pushes of axioms since all axioms are pushed independently of the execution of Alg. 1). The second proof corresponds to the trace

$$\boldsymbol{\delta}_2 = \text{PUSH}((b'_1), b_1 \vdash b_2, b'_2) \circ \cdots \circ \text{PUSH}((b'_4), b_4 \vdash tc_G, c'_G), \tag{18}$$

which has a verbalization length of $4 \times 4 = 16$. So in this case, g_1 yields the proof that has the shortest verbalization length, which is desirable since it corresponds to a shorter token sequence generated by an LLM.

However, it is easy to see that this is not always the case. Consider a third proof, structured similarly to the second but with only two proof steps. This is now the new shortest proof according to g_2. It corresponds to the trace

$$\boldsymbol{\delta}_3 = \text{PUSH}((b'_1), b_1 \vdash b_2, b'_2) \circ \text{PUSH}((b'_2), b_2 \vdash c_G, c'_G), \tag{19}$$

which has a verbalization length of $4 \times 2 = 8$, i.e., shorter than that of the first proof.

We have demonstrated that neither g_1 nor g_2 dominates the other in terms of verbalization length and that the choice may make a difference in terms of which trace is returned by the algorithm. In our experiments, however, this would not make much difference, since the arity of the inference rules in ProofWriter and DeepRD is at most 2. In fact, for the DeepRD data, the shortest proofs under the two cost functions always coincide.

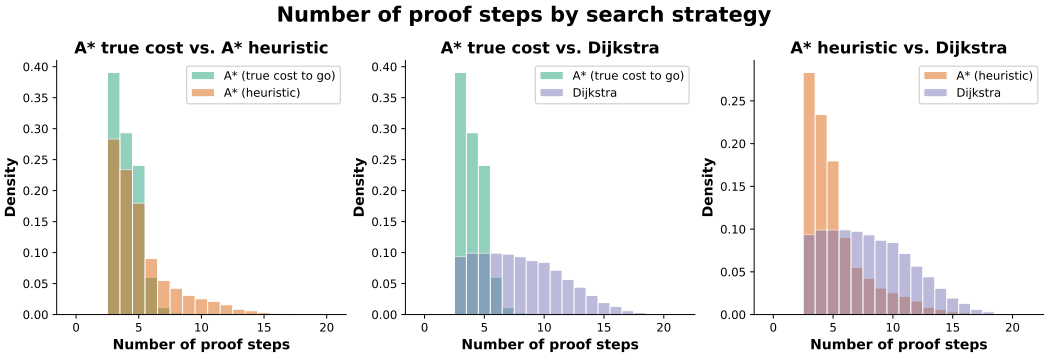

Figure 5: Comparison of number of proof steps between different search strategies. The histograms are computed based on the 9207 problems in the ProofWriter training set.

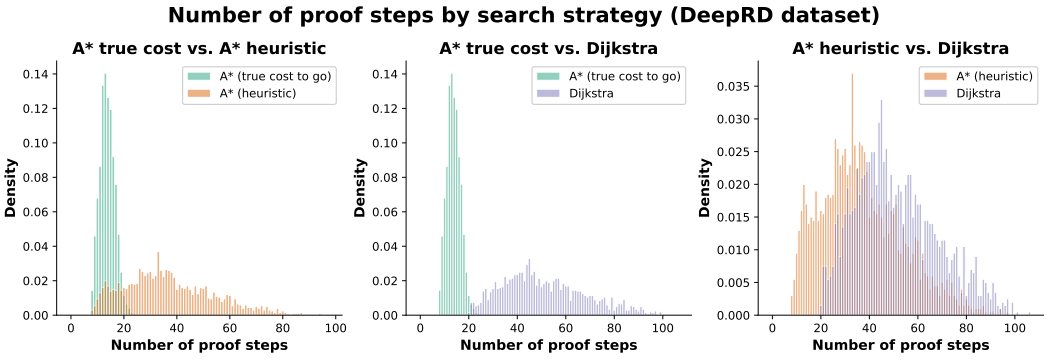

Figure 6: Comparison of number of proof steps between different search strategies. The histograms are computed based on the 2000 problems in the DeepRD training set.

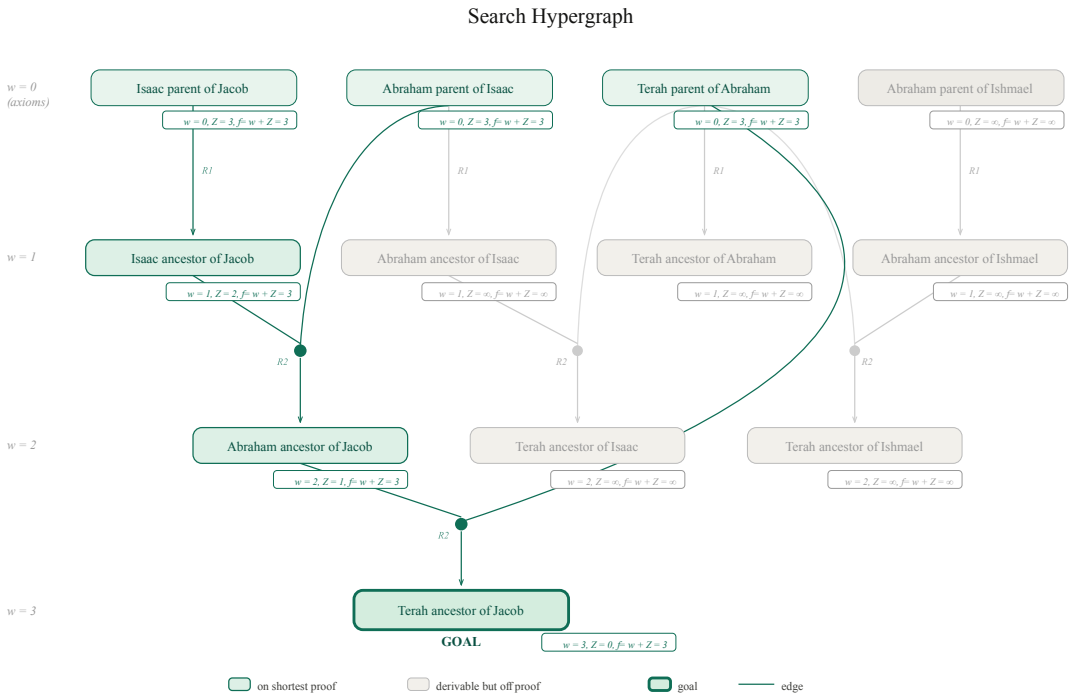

Figure 7: Hypergraph induced by the logic program from Fig. 1, showing all 12 items in the minimal Herbrand model M, arranged by weight w (i.e., depth). Each vertex is labeled with its heuristic value h under the true cost-to-go; on-path items (green) satisfy $h = w^*_G - w$. The shortest proof of ancestor(`terah`, `jacob`) traces the unique path of finite h values: $3 \to 2 \to 1 \to 0$.

D Experiment Details

We give further details on the experiments and implementations.

D.1 Efficiency Gains

We provide an analysis on the number of proof steps (defined in §3.1) taken by A* search with the dependency heuristic, A* search with the true cost-to-go, Dijkstra's algorithm for the two datasets. The results are shown in Figs. 5 and 6. Our results presented in §4 should be viewed in light of these analyses about the efficiency of the different algorithms on the two datasets.

D.2 Prompt and Verbalizations

We pass the following instruction prompt to the model:

Instruction Prompt

You will be given a logic program verbalized in natural language, including a set of rules and a set of axioms. You will also be given a goal that is provable under this logic program. The formatting will be as follows:
Rules: <rule sentence 1>. <rule sentence 2>. ... <rule sentence n>.
Axioms: <axiom sentence 1>. <axiom sentence 2>. ... <axiom sentence m>.
Goal: Prove that <goal>.
Your task is to provide a correct proof of the goal. Each proof step must follow this format (one block per step) and be delimited by two new lines:

Premises: <premise sentence 1>. ... <premise sentence k>.
Rule: <rule sentence>.
Conclusion: <conclusion sentence>.

Terminate when the conclusion of a proof step is equal to the goal using the tags <answer></answer>. Here are a few examples of programs, goals, and proofs separated by — for you to learn from:

The last sentence above is only included for the ICL experiments. For those, we fix the (program, goal)-pairs across search orders and use 10 ICL examples on ProofWriter and 5 ICL examples on DeepRD (which are longer on average). The following is an example of a verbalized (program, goal)-pair with a verbalized proof from ProofWriter:

In-Context Example

Rules: If X is blue, then X is furry. If X is nice, then X is furry. If X is blue and X is big, then X is nice. If X is cold, then X is quiet. If X is nice and X is furry, then X is cold. If gary is nice, then gary is smart. If X is cold, then X is furry. If X is cold and X is furry, then X is quiet.
Axioms: Bob is cold. Erin is nice. Gary is nice. Harry is blue.
Goal: Prove that gary is quiet.

Premises: Gary is nice.
Rule: If X is nice, then X is furry.
Conclusion: Gary is furry.

Premises: Gary is nice.
Rule: If gary is nice, then gary is smart.
Conclusion: Gary is smart.

Premises: Gary is nice. Gary is furry.
Rule: If X is nice and X is furry, then X is cold.
Conclusion: Gary is cold.

Premises: Gary is cold.
Rule: If X is cold, then X is quiet.
Conclusion: Gary is quiet.

<answer>Therefore, the goal is proven.</answer>

D.3 Training Details

For the ProofWriter data we restrict the number of generated tokens to 1024, whereas for the DeepRD data we restrict it to 2048. These maximum lengths are long enough for the longest search traces under the A* dependency heuristic and Dijkstra's algorithm. DeepSeek-V3.2 is allowed up to 2048 tokens for ProofWriter and 5000 tokens for DeepRD. We exclude ProofWriter problems whose programs contain negation. All models are trained using the Adam optimizer (Kingma & Ba, 2015).

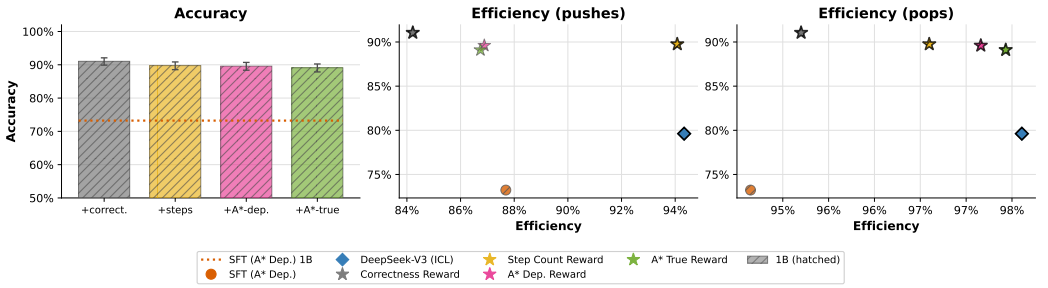

Figure 8: Results on **ProofWriter** for Llama-3.2-1B-Instruct finetuned with GRPO starting from the SFT checkpoint that was trained under the *dependency heuristic*. (Note that Fig. 3 shows results when considering the SFT checkpoint trained under the true cost-to-go).

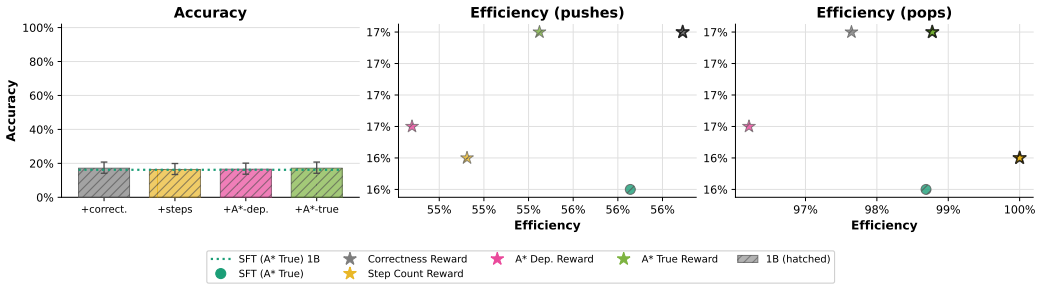

Figure 9: Results on **DeepRD** for Llama-3.2-1B-Instruct finetuned with GRPO starting from the SFT checkpoint that was trained under the *true cost-to-go*. We note that RL is not effective in this case, in which the SFT reference model is weak on the task.

For SFT, we train for three epochs, storing checkpoints after each epoch and considering the one with the lowest loss on the validation set when evaluating on the test set. For SFT we use the learning rate $2 \cdot 10^{-5}$ with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We apply cosine learning rate scheduling with a warmup ratio of 0.03. We use a batch size of 2 for the 3B model and a batch size of 4 for the 1B model. For GRPO, we train for one epoch with group size of 16, KL coefficient of 0.04, learning rate of $1 \cdot 10^{-6}$ with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. At inference, we generate solutions with greedy decoding.

E Additional Results

Figs. 8 and 9 show results for GRPO-finetuned Llama-3.2-1B-Instruct models. The results from Fig. 8 support our conclusions from Figs. 3 and 4—no reward model is overall dominating the others in terms of accuracy and efficiency. We also note that RL is unable to improve reasoning from the checkpoint trained on traces under the true cost-to-go on DeepRD (Fig. 9), which is a poor starting point; see Fig. 2. These results suggest that RL is dependent on a good reference policy.