Are Language Models Efficient Reasoners? A Perspective from Logic Programming

Andreas Opedal[α,β,*] **Yanick Zengaffinen**[α] **Haruki Shirakami**[γ,δ] **Clemente Pasti**[α]
Mrinmaya Sachan[α] **Abulhair Saparov**[ϵ] **Ryan Cotterell**[α] **Bernhard Schölkopf**[α,β]
[α]ETH Zürich [β]MPI for Intelligent Systems, Tübingen
[γ]EPFL [δ]Idiap Research Institute [ϵ]Purdue University

Abstract

Modern language models (LMs) exhibit strong deductive reasoning capabilities, yet standard evaluations emphasize correctness while overlooking a key aspect of reasoning: *efficiency*. In real-world reasoning scenarios, much of the available information is irrelevant, and effective deductive inference requires identifying and ignoring such distractions. We propose a framework for assessing LM reasoning efficiency through the lens of logic programming, introducing a simple method to align proofs written in natural language—as generated by an LM—with shortest proofs found by executing the logic program. Efficiency is quantified by measuring how well a model avoids unnecessary inference. Empirically, we construct a dataset of math word problems injected with various number of irrelevant axioms that vary in semantic overlap with the goal theorem. We find that current LMs show marked accuracy declines under such conditions—even with minimal, domain-consistent distractions—and the proofs they generate frequently exhibit detours through irrelevant inferences.[2]

1 Introduction

Large language models (LMs) appear capable of solving a wide range of tasks that rely on deductive reasoning, particularly when post-trained with reinforcement learning (Lightman et al., 2024; DeepSeek-AI, 2025) and scaled to use more compute at test time (Wang et al., 2024; Muennighoff et al., 2025; Snell et al., 2025). However, emerging findings suggest recent reasoning models often generate more tokens than necessary to solve problems, even for simple deductive tasks (Chen et al., 2025; Pu et al., 2025). Such findings point towards a key dimension of deductive reasoning that standard evaluations of LMs' reasoning abilities fail to systematically assess—*efficiency*. Indeed, more abstractly, in most real-world reasoning tasks, more information is available than is necessary to solve the problem. This spurious information is not random: it often interacts with relevant information, enabling the derivation of true but irrelevant conclusions. Crucially, it is unknown *a priori* which pieces of information will be relevant for determining whether a desired conclusion is supported by the evidence. An *efficient* solution to a problem uses only necessary information and takes as few steps as possible.

To characterize efficiency, we adopt a formalization of deductive reasoning based on *logic programming* (Kowalski, 1974). Our perspective is that logic programming provides a clean and flexible framework for reasoning within a well-understood proof system. Given a logic program—that is, a set of inference rules and axioms—a proof of some goal theorem can be viewed as a path in a hypergraph induced by the inference rules, starting from vertices corresponding to axioms and terminating at a vertex corresponding to the goal theorem. Then, the *most efficient* proof is simply a shortest such path. Using this machinery, the goal of this paper is to evaluate a *language model's* reasoning efficiency. Thus, we require an additional mechanism to bridge the gap between reasoning

[*]Please direct correspondence to andreas.opedal@inf.ethz.ch, ryan.cotterell@inf.ethz.ch, and asaparov@purdue.edu.

[2]Code for this project is available at https://github.com/rycolab/reasoning-efficiency.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

in logic programming and reasoning in natural language. To this end, we introduce the notion of a *verbalized* logic program, in which each theorem in the logic program is associated with a set of natural language strings.[3] Verbalized logic programs allow us to map the deductions performed by an LM—expressed as a natural language proof—onto deductions performed during the execution of a logic program. While numerous recent papers use number of generated tokens as a proxy for efficiency (Arora & Zanette, 2025; Han et al., 2025; Ma et al., 2025), doing so conflates inefficiency stemming from two separate sources: (1) unnecessary deduction steps and (2) verbosity in the natural language strings expressing those deduction steps. In contrast, our framework disentangles these two factors, with our paper's experiments emphasizing the former.

Empirically, we construct verbalized logic programs for grade school math word problems (GSM problems; Cobbe et al., 2021; Li et al., 2024; Zhang et al., 2024), adopting methods from Opedal et al. (2025). Our primary experimental manipulative is the injection of irrelevant axioms into these GSM programs, which yields many possible implications that are irrelevant to a goal theorem of interest. We experiment on problems that vary both in how much the information in the irrelevant axioms *overlap* with the goal theorem, as well as in *how many* such axioms are injected, generalizing existing datasets that only include a single irrelevant axiom (Shi et al., 2023; Mirzadeh et al., 2025). We first measure accuracy, showing that current LMs are less accurate on problems containing irrelevant axioms than on equivalent problems without them. This performance gap often persists even in the simplest cases, where a single irrelevant axiom from the same domain is introduced, and grows larger as more irrelevant axioms are added. We confirm that this reduction in accuracy is not due to longer inputs alone: LMs usually perform better on control problems of equal length but without irrelevant content.

Next, we map the reasoning performed by the LM onto theorems they correspond to when executing the logic program. We find that while the LMs predict most of the correct intermediate theorems for the problems where they correctly generate the goal, they are often inefficient. In particular, for GSM problems where about half of the axioms are irrelevant, more than half of the LM's predicted theorems are irrelevant too, i.e., not needed for proving the goal. The LMs are particularly inefficient when the irrelevant axioms overlap semantically with the query—for instance, when the question asks *"how many cats does Ryan have?"* and the irrelevant axioms also mention *"Ryan"* or *"cats"*. On the other hand, these results also suggest that the LMs' search procedure sometimes employ a useful heuristic based on such overlap. Our results shed some light on how LMs, albeit in natural language, perform inference.

Outline. The remainder of the paper is structured as follows: §2 situates our contributions among related work. §3 provides relevant background on logic programming and discusses how reasoning efficiency is measured relative to shortest proofs. §4 introduces verbalized logic programs and the specifics of our GSM programs. §5 presents experiments and results on how LMs reason on verbalized GSM programs with irrelevant axioms. Apps. A-D give further technical details and empirical results.

2 Related Work

Evaluating Reasoning. Most studies and benchmarks on LM reasoning evaluate correctness based on the LM's final answer (Hendrycks et al., 2021; Patel et al., 2021; Rein et al., 2024; Yu et al., 2024, *inter alia*). However, correctness of the final answer does not guarantee correctness of the proof (Lyu et al., 2023; Turpin et al., 2023). Some studies include more fine-grained reasoning evaluations by verifying LM-generated proofs (Gontier et al., 2020; Frieder et al., 2023; Nguyen et al., 2024; Wang et al., 2024; Petrov et al., 2025). While useful, many such evaluations rely either on manual scrutiny or heuristic measures of the proof's correctness. An alternative approach is to use proof assistants, e.g., Lean (de Moura & Ullrich, 2021), for formal verification (First et al., 2023; Tsoukalas et al., 2024); however, LMs may have been trained on less amounts of such data as compared to natural language. We perform an automatic evaluation by parsing the LM-generated output into proofs in logic programs.

Irrelevant Information in Reasoning Tasks. Our work relates to papers that evaluate LMs' ability to correctly solve problems with irrelevant information (or missing information; Li et al., 2025). Shi et al. (2023) create such a dataset by appending a single irrelevant statement to problems taken from GSM8k (Cobbe et al., 2021). Mirzadeh et al. (2025) seem to take a similar, albeit more manual approach; however, details presented are scarce and their dataset has not yet been made publicly available. Xu et al. (2025) incorporate irrelevant statements as part of a pipeline for generating problem variations. Anantheswaran et al. (2025) use a prompting-based method for augmenting problems with sev-

[3] Previous work have made implicit use of similar notions (e.g., Betz et al., 2021; Clark et al., 2021; Saparov & He, 2023; Morishita et al., 2023; Han et al., 2024).

eral irrelevant statements. We formalize the notion of irrelevance through logic programming and generalize previous approaches by generating problems that may have several, arbitrarily placed irrelevant axioms, which can be used together in further inference. Thus, there are many implications that are irrelevant to the goal theorem and the challenge becomes not only to generate *correct* proofs, but proofs that only contain steps that are *necessary*. By generating new problems from scratch, our approach avoids bias from memorizing the efficient solution seen during training (see, e.g., Zhang et al., 2024).

Search and Efficiency. Other studies have investigated whether and how transformers can learn search tasks (Gandhi et al., 2023, 2024; Kazemi et al., 2023; Lehnert et al., 2024; Sanford et al., 2024; Sel et al., 2024; Shah et al., 2024; Saparov et al., 2025). We are interested not only in *whether* a transformer-based LM can perform accurate search, but in how *efficient* it is. Efficiency of large (reasoning-based) LMs is a rapidly growing area of research (Sui et al., 2025), due to their often lavish use of compute (Chen et al., 2025; Pu et al., 2025). Several methods have been proposed to make reasoning more efficient (Han et al., 2025; Ma et al., 2025), e.g., by incorporating length rewards in training (Arora & Zanette, 2025; Luo et al., 2025; Team et al., 2025). While these papers focus solely on the number of tokens, we argue that it is more informative to measure efficiency based on the natural language proof, since a long output can be explained either by unnecessary inference steps or by "verbose" verbalizations of the proof. Moreover, an LM should avoid generating redundant tokens (Xia et al., 2025), but it should also not skip necessary inference steps in favor of a shorter output.

3 Logic Programming and Deductive Reasoning

This section provides relevant background on logic programming (Kowalski, 1974). It also introduces the metric we propose for scoring a proof's efficiency in §3.2.

3.1 Typed Logic Programs with Built-ins

Basic Notions. A **signature** is a 3-tuple $\Sigma = (\Sigma_{\mathrm{p}}, \Sigma_{\boldsymbol{x}}, \Sigma_{\boldsymbol{X}})$, where Σ_{p} is a set of **predicate** (or **relation**) symbols, denoted $\mathrm{p}, \mathrm{q}, \mathrm{r}, ...$; $\Sigma_{\boldsymbol{x}}$ is a set of constants, denoted $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, ...$; $\Sigma_{\boldsymbol{X}}$ is a set of variables, denoted $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, ...$[4] Every predicate is associated with an arity, which we denote using the function **arity** $\mathsf{ar}\colon \Sigma_{\mathrm{p}} \to \mathbb{N}$, specifying how many arguments it takes. Arguments to predicates are called **terms**; they can be either constants (**ground** terms) or variables (**non-ground** terms). An **atomic formula**, called an **atom**, for short, is an expression of the form $\mathrm{p}(t_1, ..., t_N)$, where $\mathrm{p} \in \Sigma_{\mathrm{p}}$ is a predicate symbol of arity $\mathsf{ar}(\mathrm{p}) = N$ and $t_1, ..., t_N \in \Sigma_{\boldsymbol{x}} \cup \Sigma_{\boldsymbol{X}}$ are all terms. The **Herbrand base** H for signature Σ is the set of all atoms that can be formed by terms in $\Sigma_{\boldsymbol{x}}$, i.e., $H = \{\mathrm{p}(t_1, ..., t_N) \mid \mathrm{p} \in \Sigma_{\mathrm{p}}, \mathsf{ar}(\mathrm{p}) = N, t_1, ..., t_N \in \Sigma_{\boldsymbol{x}}\}$.[5] Subsets of the Herbrand base $I \subseteq H$ are called **interpretations**.

Logic Programming. An **inference rule** is an expression of the form $b_1, ..., b_N \vdash h$, where $b_1, ..., b_K, h$ are atoms; $b_1, ..., b_K$ is the **body** (or **premises**) and h is the **head** (or **conclusion**). For example,

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

is an inference rule that allows us to conclude that if $\boldsymbol{X}$ is a parent of $\boldsymbol{Y}$ and $\boldsymbol{Y}$ is an ancestor of $\boldsymbol{Z}$, then $\boldsymbol{X}$ is an ancestor of $\boldsymbol{Z}$. An inference rule is called **range restricted** if each variable appearing in the conclusion h also appears in at least one atom b_k in the premise. For example, $\mathrm{p}(\boldsymbol{X}) \vdash \mathrm{q}(\boldsymbol{X})$ is range restricted, while $\mathrm{p}(\boldsymbol{X}) \vdash \mathrm{q}(\boldsymbol{X}, \boldsymbol{Y})$ is not. In this paper, we require all inference rules to be range restricted.[6] Inference rules with a null premise, i.e., where $K = 0$, and a ground conclusion, i.e., where $h \in H$, are called **axioms**. A set of axioms is denoted A. For example, $\vdash \mathrm{parent}(\boldsymbol{abraham}, \boldsymbol{isaac})$, also written $\mathrm{parent}(\boldsymbol{abraham}, \boldsymbol{isaac})$, omitting the $\vdash$ symbol, is an axiom. A **logic program** $\mathcal{P}$ over a signature Σ is a set of inference rules in which all atoms are formed by symbols in Σ. The following is an example logic program, adapted from Sterling & Shapiro (1994, §5):

$$\mathrm{parent}(\boldsymbol{terah}, \boldsymbol{abraham}) \qquad \mathrm{parent}(\boldsymbol{abraham}, \boldsymbol{ishmael}) \qquad \mathrm{parent}(\boldsymbol{abraham}, \boldsymbol{isaac})$$

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

[4] Many logic programming languages, e.g., Prolog (Colmerauer & Roussel, 1993; Körner et al., 2022), additionally have the notion of a function. Constants are then just nullary functions. Our notion of logic programming is most similar to Datalog (Vardi, 1982; Maier et al., 1984; Ceri et al., 1989), which does not.

[5] In the case that the signature additionally contains a set of function symbols Σ_f, the Herbrand base is defined as the set of all atoms that can be formed by all terms in the **Herbrand universe**, which is the smallest set U that satisfies the equation $U = \Sigma_{\boldsymbol{x}} \cup \{f(t_1, ..., t_N) \mid f \in \Sigma_f, \mathsf{ar}(f) = N, t_1, ..., t_N \in U\}$.

[6] Range restriction ensures that applying the fixpoint operator (Eq. (1)) does not create non-ground atoms.

Types and Built-ins. Our notion of logic programming additionally includes types (Abiteboul et al., 1995, §21) and built-ins (Kaminski et al., 2017), which we define here. We partition $\Sigma_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{X}}$ into T disjoint subsets, i.e., $\Sigma_{\boldsymbol{x}} = \Sigma_{\boldsymbol{x}}^1 \sqcup \cdots \sqcup \Sigma_{\boldsymbol{x}}^T$ and $\Sigma_{\boldsymbol{X}} = \Sigma_{\boldsymbol{X}}^1 \sqcup \cdots \sqcup \Sigma_{\boldsymbol{X}}^T$, respectively, and associate each subset with a **type**. These subsets are paired index-wise, i.e., $(\Sigma_{\boldsymbol{x}}^1, \Sigma_{\boldsymbol{X}}^1)$, ..., $(\Sigma_{\boldsymbol{x}}^T, \Sigma_{\boldsymbol{X}}^T)$, ensuring that the constant and variable types match. In this paper, we consider three types: (i) natural numbers, denoted $(\mathbb{N}_{\boldsymbol{x}}, \mathbb{N}_{\boldsymbol{X}})$, (ii) strings, $(\Delta_{\boldsymbol{x}}^*, \Delta_{\boldsymbol{X}}^*)$, and (iii) sets of strings, $(2_{\boldsymbol{x}}^{\Delta^*}, 2_{\boldsymbol{X}}^{\Delta^*})$. Our introduction of types is necessitated by our desire to add additional power to our notion of logic programming that is external to the language itself. Specifically, we will introduce **built-in predicates**, simply called **built-ins** through the exposition, that add various arithmetic and set-theoretic operations. We enumerate these operations:

$X_{\mathbb{Z}} + Y_{\mathbb{Z}} = Z_{\mathbb{Z}}$	(Integer Addition),	$X_{2^{\Delta^*}} \cup Y_{2^{\Delta^*}} = Z_{2^{\Delta^*}}$	(Set Union),
$X_{\mathbb{Z}} - Y_{\mathbb{Z}} = Z_{\mathbb{Z}}$	(Integer Subtraction),	$X_{2^{\Delta^*}} \cap Y_{2^{\Delta^*}} = Z_{2^{\Delta^*}}$	(Set Intersection),
$X_{\mathbb{Z}} \times Y_{\mathbb{Z}} = Z_{\mathbb{Z}}$	(Integer Multiplication),	$\lvert X_{2^{\Delta^*}} \rvert = X_{\mathbb{Z}}$	(Set Cardinality),
$X_{\mathbb{Z}} = Y_{\mathbb{Z}}$	(Integer Equality),	$X_{2^{\Delta^*}} = Y_{2^{\Delta^*}}$	(Set Equality),
$X_{\mathbb{Z}} \geq Y_{\mathbb{Z}}$	(Integer Comparison),	$\lvert X_{2^{\Delta^*}} \rvert \geq X_{\mathbb{Z}}$	(Set Cardinality Comparison).

The truth value of grounded atoms constructed from built-in predicates is evaluated *externally* to the logic program. To do so, we define the **built-in evaluator** $\mathsf{eval}\colon H \to \{\mathtt{T}, \mathtt{F}\}$, that maps all ground built-ins that evaluate to true to `T` and all ground built-ins that evaluate to false to `F`. Additionally, any element of H that is *not* a built-in evaluates to `F`. For example, $\mathsf{eval}(5{+}4{=}9) = \mathtt{T}$, $\mathsf{eval}(5{+}4{=}10) = \mathtt{F}$, and $\mathsf{eval}(\mathrm{parent}(\mathit{abraham}, \mathit{isaac})) = \mathtt{F}$. For example, to illustrate the use of built-ins in a logic program, we can extend the inference rules from the earlier example to measure the depth of the ancestor relation (e.g., parent, grandparent, great-grandparent, etc):

$$\mathrm{parent}(X_{\Delta^*}, Y_{\Delta^*}) \vdash \mathrm{ancestor}(X_{\Delta^*}, Y_{\Delta^*}, 1)$$

$$\mathrm{parent}(X_{\Delta^*}, Y_{\Delta^*}),\ \mathrm{ancestor}(Y_{\Delta^*}, Z_{\Delta^*}, X_{\mathbb{Z}}),\ X_{\mathbb{Z}} + 1 = Y_{\mathbb{Z}} \vdash \mathrm{ancestor}(X_{\Delta^*}, Z_{\Delta^*}, Y_{\mathbb{Z}})$$

Substitutions and Semantics. To assign semantics to a logic program, we require a bit more machinery. A **substitution** θ is a finite set of pairs $\{(X_m, t_m)\}_{m=1}^M$, where $X_m \in \Sigma_{\boldsymbol{X}}$, $t_m \in \Sigma_{\boldsymbol{x}} \cup \Sigma_{\boldsymbol{X}}$, $X_m \neq X_{m'}$ for all $m \neq m'$, and $X_m \neq t_m$ for all m (Sterling & Shapiro, 1994, p. 14). Additionally, a **typed substitution** is a substitution where, if $X_m \in \Sigma_{\boldsymbol{X}}^k$, then $t_m \in \Sigma_{\boldsymbol{x}}^k \cup \Sigma_{\boldsymbol{X}}^k$. We can apply a substitution θ to an atom b, denoted b/θ, e.g., $\mathrm{parent}(X, \mathit{isaac})/\{(X, \mathit{abraham})\} = \mathrm{parent}(\mathit{abraham}, \mathit{isaac})$. Let $\Theta(\mathcal{P})$ be the set of all typed substitutions under $\mathcal{P}$. We say that $b'_1, ..., b'_K \vdash h'$ is an **instantiation** of $b_1, ..., b_K \vdash h$ if there exists a substitution $\theta \in \Theta(\mathcal{P})$ such that $b'_1, ..., b'_K \vdash h' = b_1/\theta, ..., b_K/\theta \vdash h/\theta$. If an instantiation has no variables, we call it a **ground instantiation**. In addition, we say that an atom b **unifies** with b' if $\exists \theta \in \Theta(\mathcal{P}) : b/\theta = b'/\theta$. If b unifies with b' we write $b \equiv b'$. We define a logic program $\mathcal{P}$'s **fixpoint operator** $\mathrm{T}_{\mathcal{P}}\colon 2^H \to 2^H$ as

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

where $I \subseteq H$ is an interpretation. The fixpoint operator $\mathrm{T}_{\mathcal{P}}$ is **inflationary**, i.e., for every interpretation $I \subseteq H$, we have $b \in I \implies b \in \mathrm{T}_{\mathcal{P}}(I)$, and **monotone**, i.e., for every pair of interpretations $I_1, I_2 \subseteq H$, we have $I_1 \subseteq I_2 \implies \mathrm{T}_{\mathcal{P}}(I_1) \subseteq \mathrm{T}_{\mathcal{P}}(I_2)$. That $\mathrm{T}_{\mathcal{P}}$ is monotone allows us to employ least fixpoint semantics. To that end, we define the **minimal Herbrand model** as $M = \mathrm{T}_{\mathcal{P}}^*(A) \stackrel{\text{def}}{=} \bigcup_{n=0}^{\infty} \mathrm{T}_{\mathcal{P}}^n(A)$, where $\mathrm{T}_{\mathcal{P}}^n$ denotes the n-fold application of the fixpoint operator $\mathrm{T}_{\mathcal{P}}$, i.e., M is $\mathrm{T}_{\mathcal{P}}$'s least fixpoint.[7] Thus, M is the subset of the Herbrand base that is true given the axioms and inference rules in the program; we call elements of M **theorems**. Due to our inclusion of built-ins that encompass basic arithmetic operations, it is undecidable to compute M (Dantsin et al., 2001), i.e., in general, we cannot decide whether $h \in \mathrm{T}_{\mathcal{P}}^*(A)$ for an arbitrary $h \in H$.

Queries. Given a logic program $\mathcal{P}$, we are often interested in determining whether there exists a theorem in $\mathcal{P}$ that is an instantiation of a specific (possibly non-ground) atom. We refer to such atoms as **queries**.[8] For example, building on the running example drawn from Sterling & Shapiro (1994, §5), we may wish to ask whether there exists a theorem that instantiates the atom $\mathrm{ancestor}(X, \mathit{ishmael})$.

[7] Inflationarity is not needed to prove that the minimal Herbrand model M exists. Indeed, monotonicity and the fact that interpretations of the Herbrand base form a complete lattice suffice to apply Tarski's (1955) theorem, which guarantees the existence of the least fixpoint. However, inflationarity does guarantee that, as we iteratively apply $\mathrm{T}_{\mathcal{P}}$, convergence to the least fixpoint is monotone.

[8] In principle, queries may also be ground; however, only the non-ground case is of theoretical interest here, as it extends beyond what can be handled by the machinery introduced so far.

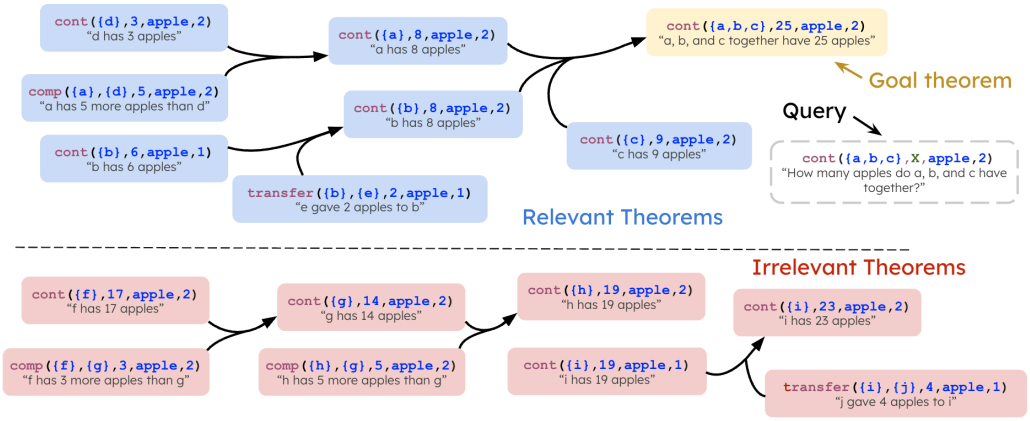

Figure 1: Example of a proof $\mathcal{P}$ of a goal theorem for the logic program presented in Table 1, with a shortest proof $\mathcal{P}^\star$ in blue and irrelevant theorems in red. We propose measuring efficiency as the size of the proof $|\mathcal{P}|$ relative to the size of the shortest proof $|\mathcal{P}^\star|$, penalizing the LM's proof for containing irrelevant theorems. We have omitted built-in predicates from this diagram for brevity.

Answering such a query amounts to finding all substitutions for X that make the atom provable from $\mathcal{P}$. When X can take on infinitely many instantiations, more sophisticated inference mechanisms—most notably, **unification** (Robinson, 1965)—are required to perform this kind of non-ground reasoning effectively. In this paper, however, we restrict attention to queries in which each variable is known *a priori* to range over a fixed, finite domain, which allows us to avoid additional complexities.

3.2 Deductive Reasoning

Hypergraphs. A **hypergraph** $\mathcal{G}$ (B-hypergraph; Gallo et al., 1993) is a tuple (V, E), where V is a set of vertices, and $E \subseteq 2^V \times V$ is a set of **hyperedges**, where a hyperedge $e = T \rightarrowtail v_h$ consists of a **tail** $T \subseteq V$, with $|T| > 0$, and a **head** $v_h \in V$. We define the size of a hypergraph as $|\mathcal{G}| \stackrel{\text{def}}{=} |V|$ where $\mathcal{G} = (V, E)$.[9] A **subhypergraph** of a hypergraph $\mathcal{G} = (V, E)$ is a hypergraph $\mathcal{G}' = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$. Given $S \subseteq V$ and $v \in V$, an (S, v)-**hyperpath** in a hypergraph $\mathcal{G} = (V, E)$ is a finite sequence of distinct hyperedges $T_1 \rightarrowtail v_{h_1}, ..., T_J \rightarrowtail v_{h_J}$ such that $v_{h_J} = v$ and for every $j \in [J]$: $T_j \subseteq S \cup \{v_{h_1}, ..., v_{h_{j-1}}\}$, i.e., each hyperedge's tail consists only of nodes that are either in the source set S or are heads of previous hyperedges in the sequence. A hyperpath generalizes the notion of a directed path in a graph, but allows each hyperedge to have multiple tail nodes that jointly produce a head node. Finding the shortest (S, v)-hyperpath in a hypergraph is analogous to context-free parsing (Klein & Manning, 2001) and can be executed in polynomial time.

Proof Forests, Proofs and Proof Efficiency. Let $\mathcal{P}$ be a logic program. A **proof forest** $(\mathcal{F}, \ell)$ in $\mathcal{P}$ is a pair where $\mathcal{F} = (E, V)$ is a hypergraph and $\ell \colon V \to H$ where H is $\mathcal{P}$'s Herbrand base (Heijltjes, 2010). Additionally, we require that, for every hyperedge $e = \{t_1, ..., t_K\} \rightarrowtail v_h \in E$, there exists a rule $(b_1, ..., b_K \vdash h) \in \mathcal{R}$ and a substitution $\theta \in \Theta(\mathcal{P})$ such that $b_1/\theta = \ell(t_1), ..., b_K/\theta = \ell(t_K)$ and $h/\theta = \ell(v_h)$. We call a proof forest $(\mathcal{F}, \ell)$ an (A, h_g)-**proof** if there exists a $(\ell^{-1}(A), \ell^{-1}(h_g))$-hyperpath in $(\mathcal{F}, \ell)$.[10] In Fig. 1, we show an example of a proof in which $h_g = \text{cont}(\{a, b, c\}, 25, \text{apple}, 2)$ in our custom logic program for math word problems (§4), together with some axioms in the program that do not contribute to the proof of the goal theorem h_g. We call an (A, h_g)-proof a **shortest proof** if it has the least number of vertices of all (A, h_g)-proofs in $\mathcal{P}$. A shortest proof can be found by forward-chaining, discussed in the subsequent paragraph. Now we turn to measuring proof efficiency. Consider an (A, h_g)-proof $\mathcal{P}$ in $\mathcal{P}$. We define the **efficiency** of $\mathcal{P}$ as $\text{EFFICIENCY}(\mathcal{P}) \stackrel{\text{def}}{=} |\mathcal{P}^\star|/|\mathcal{P}|$ where $|\mathcal{P}^\star|$ is the number of vertices in a shortest (A, h_g)-proof. In the remainder of the paper, we will refer to an axiom a as **irrelevant** if there does not exist a shortest proof $\mathcal{P}$ that contains a vertex v such that $\ell(v) = a$.

[9] We note that this is non-standard; the size of a hypergraph is more often defined as its number of hyperedges or the sum of the cardinalities of its hyperedges (Gallo et al., 1993). We use this definition to sync with the experimental setup, which we explain in §5. Future work could easily adapt our efficiency metric to other definitions.

[10] We note that the expression (A, h_g)-hyperpath is a slight abuse of notation in the case that ℓ is not injective adopted for convenience: A and h_g are a subset and an element, respectively, of $\mathcal{P}$'s Herbrand base—not of $\mathcal{R}_\mathcal{P}$'s V. In this context, by A, we refer to a set $X \subseteq \ell^{-1}(A)$ where $\ell(X) = A$, and by h_g we refer to a set $Y \subseteq \ell^{-1}(h_g)$ where $\ell(Y) = h_g$.

id	Inference Rule
(1a)	$\text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{comp}(B_{\mathcal{A}}, A_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), X_{\mathbb{N}}+Y_{\mathbb{N}}=Z_{\mathbb{N}} \vdash \text{cont}(B_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}})$
(1b)	$\text{cont}(B_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{comp}(B_{\mathcal{A}}, A_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), Z_{\mathbb{N}} \geq Y_{\mathbb{N}}, Z_{\mathbb{N}}-Y_{\mathbb{N}}=X_{\mathbb{N}} \vdash \text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}})$
(1c)	$\text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{cont}(B_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), Y_{\mathbb{N}} \geq X_{\mathbb{N}}, Y_{\mathbb{N}}-X_{\mathbb{N}}=Z_{\mathbb{N}} \vdash \text{comp}(B_{\mathcal{A}}, A_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}})$
(2a)	$\text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{transfer}(A_{\mathcal{A}}, B_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), X_{\mathbb{N}}+Y_{\mathbb{N}}=Z_{\mathbb{N}}, T_{\mathbb{N}}+1=U_{\mathbb{N}} \vdash \text{cont}(A_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, U_{\mathbb{N}})$
(2b)	$\text{cont}(B_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{transfer}(A_{\mathcal{A}}, B_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), Z_{\mathbb{N}} \geq Y_{\mathbb{N}}, Z_{\mathbb{N}}-Y_{\mathbb{N}}=X_{\mathbb{N}}, T_{\mathbb{N}}+1=U_{\mathbb{N}} \vdash \text{cont}(B_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, U_{\mathbb{N}})$
(3)	$\text{cont}(A_{\mathcal{A}_1}, Q_{\mathbb{N}_1}, E_{\Delta^*}, T_{\mathbb{N}}), ..., \text{cont}(A_{\mathcal{A}_k}, Q_{\mathbb{N}_k}, E_{\Delta^*}, T_{\mathbb{N}}), A_{\mathcal{A}_1} \cup \cdots \cup A_{\mathcal{A}_k}=B_{\mathcal{A}}, Q_{\mathbb{N}_1}+\cdots+Q_{\mathbb{N}_k}=Y_{\mathbb{N}} \vdash \text{cont}(B_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}})$ for $2 \leq k \leq K$
(4)	$\text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{rate}(A_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, F_{\Delta^*}, T_{\mathbb{N}}), X_{\mathbb{N}} \times Y_{\mathbb{N}}=Z_{\mathbb{N}} \vdash \text{cont}(A_{\mathcal{A}}, Z_{\mathbb{N}}, F_{\Delta^*}, T_{\mathbb{N}})$
(5a)	$\text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{comp}(D_{\mathcal{A}}, C_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{compeq}(A_{\mathcal{A}}, B_{\mathcal{A}}, C_{\mathcal{A}}, D_{\mathcal{A}}, E_{\Delta^*}, T_{\mathbb{N}}), X_{\mathbb{N}}+Y_{\mathbb{N}}=Z_{\mathbb{N}} \vdash \text{cont}(B_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}})$
(5b)	$\text{comp}(B_{\mathcal{A}}, A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \text{comp}(D_{\mathcal{A}}, C_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), X_{\mathbb{N}}=Y_{\mathbb{N}} \vdash \text{compeq}(A_{\mathcal{A}}, B_{\mathcal{A}}, C_{\mathcal{A}}, D_{\mathcal{A}}, E_{\Delta^*}, T_{\mathbb{N}})$
(6)	$\text{p}(A_{\mathcal{A}}, E_{\Delta^*}, T_{\mathbb{N}}, ...), \neg\text{transfer}(A_{\mathcal{A}}, B_{\mathcal{A}}, Y_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), \neg\text{transfer}(B_{\mathcal{A}}, A_{\mathcal{A}}, Z_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}}), T_{\mathbb{N}}+1 = U_{\mathbb{N}} \vdash \text{p}(A_{\mathcal{A}}, E_{\Delta^*}, U_{\mathbb{N}}, ...)$ for $\text{p} \in \{\text{cont}, \text{comp}, \text{rate}, \text{compeq}\}$

Table 1: Inference rules in $\mathcal{P}_{\text{W}}$. The symbols cont, comp, transfer, rate, compeq are predicates and we use variables $A_{\mathcal{A}}, B_{\mathcal{A}}, C_{\mathcal{A}}, C_{\mathcal{A}} \in 2^A_{\mathcal{I}}$ for agents, $E_{\Delta^*}, F_{\Delta^*} \in E_{\mathcal{I}}$ for entities, $X_{\mathbb{N}}, Y_{\mathbb{N}}, Z_{\mathbb{N}} \in \mathbb{N}^q_{\mathcal{I}}$ for quantities, and $T_{\mathbb{N}} \in \mathbb{N}^t_{\mathcal{I}}$ for timestamps. We refer to Fig. 2 for ground instantiations of the atoms shown here with corresponding example verbalizations. The symbol p in Rule (6) is a placeholder for any predicate other than transfer; because the predicates have different arities, we use "..." to denote other arguments that may be present. Rule (6) is special since it has negation (App. A.1); it is included to treat complications resulting from the timestamp in Rules (2a) and (2b)—see footnote 11—and is not used when generating shortest proofs for our experiments (App. C).

Forward Chaining. Forward chaining (Hayes-Roth et al., 1983; Poole & Mackworth, 2017) is a meta-strategy for theorem proving in logic programming that proceeds from the axioms toward the goal theorem. The process terminates once the goal theorem is proved. Pseudocode for the forward-chaining is given in Alg. 1 in App. A.2. As a meta-strategy, each instance of forward chaining defines an ordering over proof steps. Different orderings give rise to familiar search algorithms, such as depth-first search (DFS; Tarjan, 1972), breadth-first search (BFS; Moore, 1959), Dijkstra's (1959) algorithm, and heuristic-based search (Pearl, 1984) like A* (Hart et al., 1968). While DFS and BFS ignore information about the goal theorem, such information can guide search more efficiently, as exemplified by goal-aware strategies like Earley's (1970) algorithm for context-free parsing or its more general equivalent in logic programming—magic templates (Bancilhon et al., 1986; Ramakrishnan, 1991; Eisner & Blatz, 2007). In the spirit of using top-down information in proof search, in §5, we examine whether LMs make use of lexical overlap with the goal theorem as part of their internal search heuristic.

4 Evaluating Language Models on Grade School Math Word Problems

We are interested in reasoning that takes places in *natural language*, particularly as performed by LMs. To this end, we consider grade school math (GSM) word problems as empirical test domain. Such problems are commonly used for training and evaluating LMs on reasoning tasks (Cobbe et al., 2021; Patel et al., 2021). In §4.1, we introduce a family of logic programs, using the technical notions introduced in §3, that correspond to a natural class of such math word problems. We also describe a simple manner to convert text generated by an LM to a proof in such logic programs in §4.2. Finally, in §4.3 we explain how we generate problem descriptions in natural language that contain irrelevant axioms.

4.1 Modeling Math Word Problems with Verbalized Logic Programs

Logic Programming for Grade School Math Word Problems. We consider a particular family of logic programs to represent GSM problems, adapted from Opedal et al. (2023, 2025). All of the logic programs have the signature $\Sigma^{\text{W}} = (\Sigma^{\text{W}}_{\text{p}}, \Sigma^{\text{W}}_{\boldsymbol{x}}, \Sigma^{\text{W}}_{\mathcal{I}})$. The set of predicate symbols $\Sigma^{\text{W}}_{\text{p}} = \{\text{cont}, \text{comp}, \text{transfer}, \text{rate}, \text{compeq}\}$ corresponds to arithmetic concepts that occur in GSM word problems (Riley et al., 1983), e.g., cont for denoting how many entities an agent contains, comp for comparing the number of entities across multiple agents, or transfer for expressing one agent transfering entities to another. We partition $\Sigma^{\text{W}}_{\boldsymbol{x}} = 2^A_{\boldsymbol{x}} \sqcup E_{\boldsymbol{x}} \sqcup \mathbb{N}^q_{\boldsymbol{x}} \sqcup \mathbb{N}^t_{\boldsymbol{x}}$ and $\Sigma^{\text{W}}_{\mathcal{I}} = 2^A_{\mathcal{I}} \sqcup E_{\mathcal{I}} \sqcup \mathbb{N}^q_{\mathcal{I}} \sqcup \mathbb{N}^t_{\mathcal{I}}$ into the following pairs: sets of strings $(2^A_{\boldsymbol{x}}, 2^A_{\mathcal{I}})$ called **sets of agents** (*who* possesses), strings $(E_{\boldsymbol{x}}, E_{\mathcal{I}})$ called **entities** (*what* is possessed), natural numbers $(\mathbb{N}^q_{\boldsymbol{x}}, \mathbb{N}^q_{\mathcal{I}})$ called **quantities** (*how much* is possessed), and natural numbers $(\mathbb{N}^t_{\boldsymbol{x}}, \mathbb{N}^t_{\mathcal{I}})$ called **timestamps** (*time* of possession). We note that $2^A_{\boldsymbol{x}}$ is a power set of the set of agent strings $A_{\boldsymbol{x}}$, which enables us to code a state where multiple agents possess the same entity jointly. The family of logic programs all share the same inference rules, which are given in Table 1. Fig. 1 shows an example proof using these inference rules, omitting built-ins. We note that possession may change over time and is governed by the transfer predicate in Rule (2).[11]

[11]The time component requires the addition of Rule (6), which expresses that all theorems with time $T_{\mathbb{N}}$ that are *not* affected by a proof step using Rule (2) maintain their truth value at $T_{\mathbb{N}}+1$. This is an instance of the

Axioms. The rules given in Table 1 are held constant across all logic programs in our family. Indeed, what distinguishes one program from another, then, is the choice of axioms. For example, consider the axiom $\text{cont}(\{ryan\}, 5, cat, 2)$ which expresses that *"Ryan has 5 cats at time step 2"*; the predicate cont represents the semantics of possession, and $\{ryan\} \in 2^A_x$, $5 \in \mathbb{N}^q_x$, $cat \in E_x$, and $2 \in \mathbb{N}^t_x$ are all ground atoms of different types. Additionally, we only consider axiom sets $A_W \subset \overline{H}$ that have the following property: We call an interpretation $I \subseteq \overline{H}$ of a logic program in our family (Table 1) **numerically consistent** if there do *not* exist two substitutions $\theta = \{(A_\mathcal{A}, t_a), (X_\mathbb{Z}, t), (E_{\Delta^*}, t_e), (T_\mathbb{N}, t_t)\}$ and $\theta' = \{(A_\mathcal{A}, t_a), (X_\mathbb{Z}, t'), (E_{\Delta^*}, t_e), (T_\mathbb{N}, t_t)\}$ such that $t \neq t'$ and both $\text{cont}(A_\mathcal{A}, X_\mathbb{N}, E_{\Delta^*}, T_\mathbb{N})/\theta, \text{cont}(A_\mathcal{A}, X_\mathbb{N}, E_{\Delta^*}, T_\mathbb{N})/\theta' \in \text{T}^*_{\mathcal{P}_W}(I)$. This ensures that the minimal Herbrand model does not contain contradictory pairs such as $\text{cont}(\{ryan\}, 5, cat, 2)$ and $\text{cont}(\{ryan\}, 4, cat, 2)$. For example, the following set of axioms is numerically consistent: $A_W = \{\text{cont}(\{ryan\}, 5, cat, 2), \text{comp}(\{eleanor\}, \{ryan\}, 3, cat, 2)\}$, expressing that *"Ryan has 5 cats"* and that *"Eleanor has 3 more cats than Ryan"*. If we were to, e.g., include the two axioms $\{\text{cont}(\{andreas\}, 7, cat, 2), \text{comp}(\{eleanor\}, \{andreas\}, 3, cat, 2)\}$, we would obtain a numerically inconsistent axiom set. Inference in our family of logic programs is decidable under numerically consistent axiom sets. See App. B for details.

4.2 Language Modeling and Proofs in Natural Language

Language Modeling. We give a brief formal introduction to language modeling. Let Γ be an alphabet of tokens and Γ^* be the set of all strings over Γ, its Kleene closure. We write $\boldsymbol{w} \in \Gamma^*$ for a string, w_t for the token at the t^{th} position in $\boldsymbol{w} = w_1 \cdots w_T$, and $|\boldsymbol{w}| = T$ for the number of tokens in $\boldsymbol{w}$, i.e., its length. A **language model** (**LM**) p is a probability distribution on Γ^*. Let $\text{EOS} \notin \Gamma$ be a distinguished symbol denoting the end of a token string. The probability of a string $\boldsymbol{w}$ can be written autoregressively as $p(\boldsymbol{w}) = \overrightarrow{p}(\text{EOS} \mid \boldsymbol{w}) \prod_{t=1}^{|\boldsymbol{w}|} \overrightarrow{p}(w_t \mid \boldsymbol{w}_{<t})$, where $\overrightarrow{p}(\cdot \mid \boldsymbol{c})$ is a probability distribution over $\overline{\Gamma} \stackrel{\text{def}}{=} \Gamma \cup \{\text{EOS}\}$ conditioned on the context $\boldsymbol{c} \in \Gamma^*$.

Verbalized Logic Programs. To bridge reasoning in formal proof systems to reasoning in natural language, we introduce the idea of a **verbalized logic program**. Given a logic program $\mathcal{P}$ with negation (App. A.1), let $\overline{H}$ be its extended Herbrand base. We associate each atom $b \in \overline{H}$ with a set of natural language strings. To that end, we define a **verbalizer** $\nu_\mathcal{P} \colon \overline{H} \to 2^{\Gamma^*}$, where each $\nu_\mathcal{P}(b)$ is a disjoint set. For every $b \in \overline{H}$, the set $\nu_\mathcal{P}(b)$ represents the various ways in which the meaning of b can be expressed in natural language. In this paper, we take a straightforward approach. For each $b \in \overline{H}$, we construct a finite set $G_b \subset \Gamma^*$. Moreover, we enforce disjointness, i.e., $G_{b_2} \cap G_{b_1} = \varnothing$ for $b_1, b_2 \in \overline{H}$, and that, for all $b \in H$, each string in G_b ends in a distinguished separator symbol—in our case a period "."—that appears nowhere else in the string. These assumptions allow for trivial linear-time parsing of natural language text into a sequence of atoms in the verbalized logic programs. In practice, we generate the strings in each G_b with a series of hand-written templates. We give examples in Fig. 2.

Grounded Atom	Verbalization
$\text{cont}(\{alice\}, 3, apple, 1)$	*"Alice has 3 apples."*
$\text{cont}(\{alice, bob\}, 8, apple, 1)$	*"Alice and Bob have 8 apples combined."*
$\text{comp}(\{bob\}, \{alice\}, 2, apple, 1)$	*"Bob has 2 more apples than Alice."*
$\text{transfer}(\{alice\}, \{bob\}, 2, apple, 1)$	*"Bob gave 2 apples to Alice."*
$\text{rate}(\{alice\}, 4, basket, apple, 1)$	*"Each of Alice's baskets contains 4 apples."*
$3+2=5$	Not verbalized

Figure 2: Examples of verbalized grounded atoms, i.e., elements of $\nu_{\mathcal{P}_W}(b)$ for some $b \in H$. Built-ins are not verbalized other than as part of the conclusion; App. D.2 shows how this was done in our prompt. We additionally note that time is not verbalized directly but implicitly through the use of a past form depending on context.

4.3 Generating Problems with Irrelevant Axioms

Sampling Axiom Sets. We introduce a simple sampling algorithm, presented and analyzed in App. C, that samples a ground goal theorem $h_g \equiv \text{cont}(A_\mathcal{A}, X_\mathbb{N}, E_{\Delta^*}, T_\mathbb{N})$ and a shortest proof of h_g under the rules $\mathcal{R}_W$ from Table 1. By construction, the axioms A_W in the shortest proof are numerically consistent (§4.1) and *all* axioms in A_W will be used in the proof of h_g in the program $\mathcal{P}_W = \mathcal{R}_W \sqcup A_W$; we denote this dependency by $A_W(h_g)$.[12] That is, $A_W(h_g)$ contains no irrelevant

frame problem (McCarthy & Hayes, 1969; Sandewall, 1972; Hanks & McDermott, 1987). To express this rule, we introduce negation into the logic program; see App. A.1 for background on negation in logic programming. The only negated predicate in Table 1 is ¬transfer; because ¬transfer only appears in the body of an inference rule, the program is trivially stratified (App. A.1). That is ¬transfer atoms can not be proved true, but we assume ¬transfer atoms if the corresponding transfer atom is not known to be true from the axioms.

[12] $A_W(h_g)$ is the unique smallest set of axioms (Lemma 1) needed to prove h_g, justifying the function notation.

Model	Base	w/ Axiom		w/ Tree		w/ Multiple Trees	
		Irrelevant	Control	Irrelevant	Control	Irrelevant	Control
Llama-3.1-8B	65.0	60.8	54.0	52.6	58.6	41.0	52.4
	$(-4.3, 4.1)$	$(-2.2, 2.1)$	$(-4.4, 4.3)$	$(-2.2, 2.2)$	$(-4.4, 4.2)$	$(-2.2, 2.2)$	$(-4.4, 4.3)$
Qwen2.5-Math-7B	52.8	43.4	48.4	35.6	40.2	23.5	30.6
	$(-4.4, 4.3)$	$(-2.2, 2.2)$	$(-4.4, 4.4)$	$(-2.1, 2.2)$	$(-4.2, 4.4)$	$(-1.8, 1.9)$	$(-3.9, 4.2)$
QwQ-32B	63.4	62.2	76.2	58.7	72.2	50.1	65.4
	$(-4.3, 4.1)$	$(-2.2, 2.1)$	$(-3.9, 3.5)$	$(-2.2, 2.1)$	$(-4.1, 3.7)$	$(-2.2, 2.2)$	$(-4.3, 4.0)$
DeepSeek-R1	99.4	98.2	98.4	98.0	98.6	97.6	98.6
	$(-1.1, 0.4)$	$(-0.7, 0.5)$	$(-1.5, 0.8)$	$(-1.1, 0.9)$	$(-0.8, 0.6)$	$(-1.1, 0.9)$	$(-1.5, 0.7)$

Table 2: Average answer accuracy (%) with 95% confidence interval (CI) for base problems augmented with different degrees of irrelevance (one irrelevant axiom, one irrelevant tree, and multiple irrelevant trees; §4.3). The control problems are of the same length as the corresponding problems that have irrelevant axioms, with the difference being that all axioms are relevant. The CIs are Wilson (1927) score intervals. Note that adding irrelevant axioms degrades performance across all models.

axioms. However, to produce logic programs that *do* contain irrelevant axioms, we do the following. We sample an additional M distinct distractor theorems $\{\widetilde{h}_m\}_{m=1}^{M}$. For each distractor theorem $\widetilde{h}_m$, we again apply our axiom sampling procedure to generate distractor axioms $\widetilde{A}_m(\widetilde{h}_m)$. Importantly, we are able to show that $A_{\text{W}}(h_g)$ remain on a shortest proof of h_g *even* in the case that we consider the augmented logic program $\widetilde{\mathcal{P}}_{\text{W}} = \mathcal{R}_{\text{W}} \sqcup A_{\text{W}}(h_g) \sqcup \widetilde{A}_1(\widetilde{h}_1) \sqcup \cdots \sqcup \widetilde{A}_M(\widetilde{h}_M)$; see App. C.4.

Structural Overlap. In our experiments, we vary the size of the irrelevant axiom sets $\widetilde{A} = \widetilde{A}_1(\widetilde{h}_1) \sqcup \cdots \sqcup \widetilde{A}_M(\widetilde{h}_M)$ as follows: (i) a single irrelevant axiom (**w/ axiom**), where $M = 1$, $|\widetilde{A}_1(\widetilde{h}_1)| = 1$, and $\widetilde{A}_1(\widetilde{h}_1) = \{\widetilde{h}_1\}$ (i.e., the distractor theorem is trivial); (ii) a single irrelevant tree (**w/ tree**), where $M = 1, |\widetilde{A}_1(\widetilde{h}_1)| > 1$, and $\widetilde{h}_1$ has a non-trivial proof; (iii) multiple irrelevant trees (**w/ multiple trees**), where we set $M = 3$, $|\widetilde{A}_m(\widetilde{h}_m)| > 1$ for $m \in \{1, 2, 3\}$, and $\widetilde{h}_1, \widetilde{h}_2, \widetilde{h}_3$ all have non-trivial proofs.

Agent and Entity Overlap. We additionally control for overlap between the set of agents, i.e., elements of 2^{A}_{x}, and the entities, i.e., elements of E_x, of the goal theorem $h_g \equiv \text{cont}(A_{\mathcal{A}}, X_{\mathbb{N}}, E_{\Delta^*}, T_{\mathbb{N}})$, and those present in $\widetilde{A}$. Intuitively, for a query asking how many telescopes *"Bernhard"* has, we should make use of the information in *"telescope"* and *"Bernhard"* when deciding whether to take a certain deduction step. By constructing irrelevant axioms that also mention *"telescope"* and/or *"Bernhard"*, it becomes harder to distinguish what is relevant and what is not. We distinguish four cases: (i) neither the set of agents nor the entity occur in $\widetilde{A}$ (**no overlap**), (ii) the agent does not occur in $\widetilde{A}$ but the entity does (**entity overlap**), (iii) the entity does not occur in $\widetilde{A}$ but the set of agents does (**agent overlap**), and (iv) entity overlap in which the agents occurring in $\widetilde{A}$ have *lexical* overlap with the set of agents (**agent and entity overlap**), e.g., if `bernhard` is in h_g then $\widetilde{A}$ contains agents like `bernhard's_student` or `bernhard's_son`. Entities that do *not* overlap are always made topically related, e.g., if A_{W} contains axioms with `telescope`, then $\widetilde{A}$ may contain `binocular`.

5 Experiments

We use proofs generated from the family of verbalized logic programs introduced in §4 to help understand how LMs reason about GSM problems. Our primary experimental manipulative is irrelevant axioms with respect to a goal theorem introduced into a logic program, which allows us to analyze how LMs fare in the face of irrelevant axioms. We generate 500 problems with varying structural, agent and entity overlap, as discussed above. See App. D.1 for more details on the make-up of the dataset. We refer to the problems *without* any irrelevant axioms as **base problems**. We additionally generate control problems that have the same number of axioms as the problems with irrelevant axioms, except that all axioms are relevant. Their shortest proofs contain the base problem's shortest proof as a subproof. This controls for the possible confounder of problem length (see, e.g., Leeb et al., 2025). We consider both non-ground queries, corresponding to questions like *"How many drones does Yanick have?"*,[13] and ground queries, corresponding to questions like *"Show that Yanick has 5 drones."*.

[13] Assuming numerical consistency ensures there exists at most one ground atom in the minimal Herbrand model that unifies with the non-ground query. This ensures that forward chaining would halt in finite time.

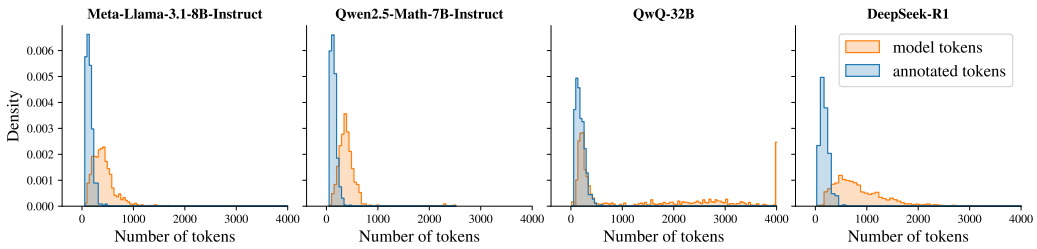

Figure 3: Number of tokens in the natural language annotation of the shortest proof as compared to model outputs for problems with three irrelevant trees ($\widetilde{A}_1 \sqcup \widetilde{A}_2 \sqcup \widetilde{A}_3$). The plot includes problems for which the model gave the correct final answer; this is why the density for annotated tokens differs between the models. We observe that LMs often use more tokens than necessary.

Language Models. We use one "vanilla" LM and three reasoning LMs in our experiments: Llama-3.1-8B-Instruct (Llama Team, 2024), Qwen2.5-Math-7B-Instruct (Yang et al., 2024), Qwen's reasoning model QwQ-32B, and DeepSeek-R1 (DeepSeek-AI, 2025). We generate strings using ancestral sampling, restricting the context length to 4000 tokens.

Prompting. Our experimental design is based on in-context learning (Brown et al., 2020). Specifically, we use five fixed in-context examples of shortest proofs to expose the LM to proofs in our verbalized logic programs. The verbalized proofs are ordered under a DFS traversal of the theorems in the proof, such that the axioms are popped in the same order they occur in the verbalized text. See App. D.2 for the prompt.

5.1 Addition of Irrelevant Axioms and Answer Accuracy

We begin by analyzing how irrelevant axioms influence an LM's ability to generate the correct goal theorem for non-ground queries. We employ the two-step prompting strategy given by Kojima et al. (2022). In the first step, the model is prompted to produce a natural-language proof outlining its reasoning process. This natural-language process is then mapped to a proof in the verbalized logic program. In the second step, we prompted the LM a second time—conditioned on the proof it generated—to produce the goal theorem. In Table 2, we report model accuracy in generating the correct goal theorem. We observe that even a single irrelevant axiom reduces model performance, particularly for Llama-3.1 and Qwen2.5. Performance degrades further as additional irrelevant axioms are introduced. The performance of the most capable model, DeepSeek-R1, is nearly saturated at perfect accuracy, though slight decreases are still observed when irrelevant axioms are included. Across models, accuracy on problems containing irrelevant axioms is almost always lower than on the corresponding control examples, suggesting that irrelevance has a substantial effect on accuracy beyond what can be explained by longer problem statements. The only exception is Llama-3.1, whose performance on problems with one irrelevant axiom exceeds that of the corresponding control. QwQ-32B stands out as an outlier: for this model, performance on the control problems is significantly higher than on the original base problems. In Fig. 7 (App. D.4), we present results stratified by agent and entity overlap (§4.3). A consistent pattern emerges: such overlap between the goal and irrelevant axioms makes solving the problem more difficult. Compared to no overlap, performance drops with both kinds of overlap, suggesting both serve as heuristics during search. While drops are typically larger for agent overlap than for entity overlap, we note that this could be partially due to the entities being topically related (§4.3). In the following subsection, we analyze the proofs in greater detail to further illuminate these effects.

5.2 Addition of Irrelevant Axioms and Efficiency

In Fig. 3 we plot the empirical distribution over the number of tokens in the model's output and compare it to the number of tokens in the natural language annotation of the shortest proof, taking only the proofs that concluded at the correct goal theorem. This analysis, as well as those in the remainder of this section, are done on the problems with multiple irrelevant trees (i.e., the kind of structural overlap with the most axioms; §4.3). We observe that all models often use more tokens than are in the annotations, suggesting that they use more compute than necessary to prove goal theorems. This is consistent with findings on reasoning models (§2), but holds also for the Llama model. However, these results do not confirm that the models generate irrelevant theorems—they might just be more verbose than our annotations. This leads us into our efficiency analysis in line with the technical exposition in §3.2.

		Non-ground Queries				Ground Queries			
		None	Entity	Agent	Both	None	Entity	Agent	Both
Llama-3.1-8B	Efficiency	60.2	43.5	47.5	34.0	45.3	34.9	29.9	23.6
	Exact matches	15	3	4	0	11	9	7	11
	Efficiency (non-axioms)	71.8	50.3	63.0	44.2	57.6	41.3	42.9	37.1
	Correct Theorems	84.5	82.9	86.0	85.4	76.0	77.8	72.1	69.8
Qwen2.5-Math	Efficiency	43.7	32.4	34.4	31.1	36.8	30.6	29.3	25.3
	Exact matches	8	0	1	0	10	1	2	1
	Efficiency (non-axioms)	57.3	40.1	51.5	46.2	50.5	38.1	41.1	36.0
	Correct Theorems	75.1	73.7	74.3	73.6	62.5	63.4	57.4	56.6

Table 3: We report efficiency, number of exact matches (out of 500), and efficiency restricted to non-axioms between the theorems parsed from the output generated by LMs and the shortest, most-efficient proof. Results are stratified by type of query and type of overlap between agents and entities present in the query and the irrelevant axioms. The numbers are computed based on the problems for which the model got the correct final answer. The low efficiency scores show that LMs often produce irrelevant theorems when successfully proving a goal theorem. The differences in efficiency scores across datasets of different overlaps are significant ($p < 0.001$) according to one-way ANOVA analyses. Lastly, "Correct Theorems" shows the proportion of the theorems in the shortest proof that the LM predicted.

Efficiency Evaluation. This analysis is performed for Llama-3.1-8B-Instruct and Qwen2.5-Math-7B-Instruct since only those models followed the required formatting (§4.2);[14] however, we perform a more crude analysis based on only the arithmetic expressions for the other two models in App. D.4. We report the efficiency metric presented in §3.2 for the problems where the LM generated the correct goal theorem, comparing against the shortest proof generated with our method (App. C).[15] We omit built-ins from the efficiency analysis since those are not verbalized. Additionally, we note that it might be the case that all irrelevant steps the models take are axioms; the LMs may simply be stating that some axioms are irrelevant to the query. We therefore also consider a metric in which only non-axiom theorems are counted. In App. D.4 we provide an additional analysis on search order.

Efficiency for Non-ground Queries. The main results are shown in §5.2 (non-ground queries), with scores stratified by agent and entity overlap. The efficiency scores are far from 100%, meaning that the models predict several theorems beyond the required ones present in the shortest proof. Additionally, we observe that the efficiency scores vary significantly across the type of overlap ($p < 0.001$), so we conclude that lexical information in the query has a substantial effect on proof planning. We finally comment on the results that only consider non-axioms, presented at the last row in §5.2. We again observe efficiency scores that are considerably below 100%, showing that the LMs prove theorems that are irrelevant to the query. App. D.3 gives an example where Llama-3.1-8B proves irrelevant theorems.

Comparison to Ground Queries. We compare the performance to the same problems when presented with ground queries. Queries tend to be non-ground in the GSM domain (Riley et al., 1983; Cobbe et al., 2021). Since LMs are heavily influenced by training data, we therefore expect them to perform better on non-ground queries. Our results on the same verbalized logic programs as before, but with ground queries, are presented on the right-hand side of §5.2. We observe lower efficiency scores, suggesting that LMs are indeed worse at proving ground theorems in this domain.

6 Conclusion

This paper provided a framework based on logic programming for studying deductive reasoning in language models, with a particular emphasis on efficiency. This framework enables us to disentangle efficiency due to generating irrelevant theorems from efficiency due to verbose natural language verbalizations of those theorems. We applied this framework to empirically investigate how language models perform reasoning on math word problems that have many irrelevant axioms. We found that introducing irrelevant axioms into reasoning problems leads to significantly lower answer accuracies for most models—even when controlling for problem length—and proofs that exhibit frequent detours through irrelevant theorems. Our work highlights the need to improve models in terms of reasoning efficiency, as well as the advantages to viewing *deductive* reasoning through the lens of logic programming.

[14] We manually verified parsing accuracy on the theorems generated by the models for a subset of 20 randomly sampled examples. An additional class representing that there is no match with any annotated theorem in the proof is included as well. The parser predicted the correct (or correctly predicted no) match in $394/397 = 99.2\%$ of the theorems for Llama-3.1-8B-Instruct, and $381/384 = 99.2\%$ of the theorems for Qwen2.5-Math-7B-Instruct.

[15] LMs often generated the same theorems multiple times. We chose to ignore such duplicates in the evaluation.

Acknowledgments and Disclosure of Funding

We thank Juan Luis Gastaldi for useful discussion and criticisms. Andreas Opedal acknowledges funding from the Max Planck ETH Center for Learning Systems.

References

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1995. ISBN 0201537710. URL https://dl.acm.org/doi/10.5555/551350.

Ujjwala Anantheswaran, Himanshu Gupta, Kevin Scaria, Shreyas Verma, Chitta Baral, and Swaroop Mishra. Cutting through the noise: Boosting LLM performance on math word problems. In *Proceedings of the ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=VnPYbWQjz7.

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.

François Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman. Magic sets and other strange ways to implement logic programs. In *Proceedings of the 5th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*, pp. 1–15, 1986. URL https://doi.org/10.1145/6012.15399.

Gregor Betz, Christian Voigt, and Kyle Richardson. Critical thinking for language models. In Sina Zarrieß, Johan Bos, Rik van Noord, and Lasha Abzianidze (eds.), *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pp. 63–75, Groningen, The Netherlands (online), June 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.iwcs-1.7/.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

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 Engingeering*, 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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Alain Colmerauer and Philippe Roussel. The birth of Prolog. *SIGPLAN Not.*, 28(3):37–52, March 1993. doi: 10.1145/155360.155362. URL https://doi.org/10.1145/155360.155362.

Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001. URL https://doi.org/10.1145/502807.502810.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. URL https://link.springer.com/chapter/10.1007/978-3-030-79876-5_37.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

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.

Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970. doi: 10.1145/357980.358005. URL https://doi.org/10.1145/362007.362035.

Jason Eisner and John Blatz. Program transformations for optimization of parsing algorithms and other weighted logic programs. In *Proceedings of the Conference on Formal Grammar*, 2007. URL http://cs.jhu.edu/~jason/papers/#eisner-blatz-2007.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2023, pp. 1229–1241, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703270. doi: 10.1145/3611643.3616243. URL https://doi.org/10.1145/3611643.3616243.

Simon Frieder, Luca Pinchetti, Chevalier Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. Mathematical capabilities of ChatGPT. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 27699–27744. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/58168e8a92994655d6da3939e7cc0918-Paper-Datasets_and_Benchmarks.pdf.

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, Dorsa Sadigh, and Noah Goodman. Strategic reasoning with language models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=MUtbsFRZwI.

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.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Chris Pal. Measuring systematic generalization in neural proof generation with transformers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22231–22242. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fc84ad56f9f547eb89c72b9bac209312-Paper.pdf.

Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 5(4):4–21, 2018. URL https://dl.acm.org/doi/10.1145/3242953.3242964. A concise tutorial overview on decision procedures and complexity of Presburger arithmetic.

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/.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(87)90043-9. URL https://www.sciencedirect.com/science/article/pii/0004370287900439.

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.

Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1983. URL https://doi.org/10.1109/PROC.1985.13159.

Willem Heijltjes. Classical proof forestry. *Annals of Pure and Applied Logic*, 161(11):1346–1366, 2010. ISSN 0168-0072. doi: https://doi.org/10.1016/j.apal.2010.04.006. URL https://www.sciencedirect.com/science/article/pii/S0168007210000473. Special Issue: Classical Logic and Computation (2008).

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Mark Kaminski, Bernardo Cuenca Grau, Egor V. Kostylev, Boris Motik, and Ian Horrocks. Foundations of declarative data analysis using limit Datalog programs. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 1123–1130. AAAI Press, 2017. ISBN 9780999241103. URL https://www.ijcai.org/proceedings/2017/0156.pdf.

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/.

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/.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Robert Kowalski. Predicate logic as programming language. In *IFIP congress*, volume 74, pp. 569–544, 1974. URL https://www.doc.ic.ac.uk/~rak/papers/IFIP%2074.pdf.

Keito Kudo, Yoichi Aoki, Tatsuki Kuribayashi, Shusaku Sone, Masaya Taniguchi, Ana Brassard, Keisuke Sakaguchi, and Kentaro Inui. Think-to-talk or talk-to-think? When LLMs come up with an answer in multi-step reasoning. *arXiv preprint arXiv:2412.01113*, 2024. URL https://arxiv.org/abs/2412.01113.

Philipp Körner, Michael Leuschel, João Barbosa, Vitor Santos Costa, Veronica Dahl, Manuel V. Hermenegildo, Jose F. Morales, Jan Wielemaker, Daniel Diaz, Salvador Abreu, and Giovanni. Ciatto. Fifty years of Prolog and beyond. *Theory and Practice of Logic Programming*, 22(6): 776–858, 2022. URL https://doi.org/10.1017/S1471068422000102.

Felix Leeb, Zhijing Jin, and Bernhard Schölkopf. Causality can systematically address the monsters under the bench(marks). *arXiv preprint arXiv:2502.05085*, 2025. URL https://arxiv.org/abs/2502.05085.

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? *arXiv preprint arXiv:2503.22674*, 2025. URL https://arxiv.org/abs/2503.22674.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2961–2984, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.163. URL https://aclanthology.org/2024.acl-long.163/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

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

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.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In Jong C. Park, Yuki Arase, Baotian Hu, Wei Lu, Derry Wijaya, Ayu Purwarianti, and Adila Alfa Krisnadhi (eds.), *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 305–329, Nusa Dua, Bali, November 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.ijcnlp-main.20. URL https://aclanthology.org/2023.ijcnlp-main.20/.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. CoT-valve: Length-compressible chain-of-thought tuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.300. URL https://aclanthology.org/2025.acl-long.300/.

David Maier, Jeffrey D. Ullman, and Moshe Y. Vardi. On the foundations of the universal relation model. *ACM Trans. Database Syst.*, 9(2):283–308, June 1984. ISSN 0362-5915. doi: 10.1145/329.318580. URL https://doi.org/10.1145/329.318580.

John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and Donald Michie (eds.), *Machine Intelligence 4*, pp. 463–502. Edinburgh University Press, 1969. URL https://philpapers.org/rec/MCCSPP.

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.

Edward F. Moore. The shortest path through a maze. *Proceedings of an International Symposium on the Theory of Switching*, pp. 285–292, 1959. URL https://books.google.ch/books?id=IVZBHAAACAAJ.

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.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=LdH0vrgAHm.

Minh-Vuong Nguyen, Linhao Luo, Fatemeh Shiri, Dinh Phung, Yuan-Fang Li, Thuy-Trang Vu, and Gholamreza Haffari. Direct evaluation of chain-of-thought in multi-hop reasoning with knowledge graphs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2862–2883, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.168/.

Andreas Opedal, Niklas Stoehr, Abulhair Saparov, and Mrinmaya Sachan. World models for math story problems. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 9088–9115, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.579. URL https://aclanthology.org/2023.findings-acl.579/.

Andreas Opedal, Haruki Shirakami, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. MathGAP: Out-of-distribution evaluation on problems with arbitrarily complex proofs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5ck9PIrTpH.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? 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. 2080–2094, Online, June 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.naacl-main.168.

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA, 1984. ISBN 0-201-05594-5. URL https://dl.acm.org/doi/book/10.5555/525.

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.

David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, USA, 2nd edition, 2017. ISBN 110719539X. URL https://doi.org/10.1017/9781009258227.

Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I^{er} congrès de Mathématiciens des pays Slaves, Varsovie 1929*, pp. 92–101, 1929. URL https://books.google.ch/books?id=7agKHQAACAAJ. In German.

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.

Raghu Ramakrishnan. Magic templates: A spellbinding approach to logic programs. *The Journal of Logic Programming*, 11(3&4):189–216, 1991. URL https://doi.org/10.1016/0743-1066(91)90026-L.

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.

Mary Riley, James Greeno, and Joan Heller. *Development of Children's Problem-Solving Ability in Arithmetic*, pp. 153–196. Learning Research and Development Center, University of Pittsburgh, 01 1983. ISBN 978-0122847806. URL https://eric.ed.gov/?id=ED252410.

John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965. doi: 10.1145/321250.321253. URL https://doi.org/10.1145/321250.321253.

Erik Sandewall. An approach to the frame problem and its implementation. *Machine intelligence*, 7 (195-204):11–19, 1972. URL https://www.ida.liu.se/ext/caisor/archive/1972/001/caisor-1972-001.pdf.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=AfzbDw6DSp.

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.

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.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Huai hsin Chi, Nathanael Scharli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, 2023. URL https://openreview.net/forum?id=JSZmoN03Op.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

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.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=HvoG8SxggZ.

Hisao Tamaki and Taisuke Sato. Unfold/Fold transformation of logic programs. In *Proceedings of the International Conference on Logic Programming*, 1984. URL https://ci.nii.ac.jp/naid/10000035006/.

Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2): 146–160, 1972. doi: 10.1137/0201010. URL https://doi.org/10.1137/0201010.

Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955. URL https://msp.org/pjm/1955/5-2/pjm-v5-n2-p11-s.pdf.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, January 2025. URL https://doi.org/10.48550/arXiv.2501.12599.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the Putnam Mathematical Competition. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=ChKCF75Ocd.

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.

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.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1927.10502953.

Wilson Wu, John Xavier Morris, and Lionel Levine. Do language models plan ahead for future tokens? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BaOAvPUyBO.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39: 27723–27730, 04 2025. doi: 10.1609/aaai.v39i26.34987. URL https://ojs.aaai.org/index.php/AAAI/article/view/34987.

Xinnuo Xu, Rachel Lawrence, Kshitij Dubey, Atharva Pandey, Risa Ueno, Fabian Falck, Aditya V. Nori, Rahul Sharma, Amit Sharma, and Javier Gonzalez. RE-IMAGINE: Symbolic benchmark synthesis for reasoning evaluation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=QJPlODWajD.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Vishnu Raja, Charlotte Zhuang, Dylan Z Slack, Qin Lyu, Sean M. Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. A careful examination of large language model performance on grade school arithmetic. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=RJZRhMzZzH.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Compare the claims with the results presented in §5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in an ad-hoc manner throughout the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See App. B and App. C.4 for proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have included as much relevant details in the main text as possible. Further details on data generation can be found in App. C and details on the experiments in App. D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All code is available on our GitHub repository.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified everything we believe to be relevant, along with all details reviewers identified as missing.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Please refer to the tables and figures.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The compute required to reproduce our experiments is not outside of the ordinary typically accessed by academic labs. Inference on all models we consider is also available through APIs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We foresee no ethical concerns about our work.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: Foundational research that is not tied to particular applications.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are properly cited and used according to license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the data and code that generated it on GitHub.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Further Background

A.1 Negation in Logic Programming

It is also useful to introduce a notion of negation into logic programming. We denote negation by $\neg$. To accommodate negation, we introduce the **extended Herbrand base** $\overline{H} \stackrel{\text{def}}{=} H \cup \{\neg h \mid h \in H\}$. We call an interpretation $I \subseteq \overline{H}$ **consistent** iff $b \in I \implies \neg b \notin I$. Furthermore, a logic program $\mathcal{P}$ is called **consistency-preserving** if I is consistent implies $\mathrm{T}_{\mathcal{P}}(I)$ is consistent. One simple way to check whether a logic program is consistency-preserving is to inspect the program's dependency structure. Construct the **predicate dependency graph** G as follows. Create a vertex (p) for every distinct predicate symbol p appearing in $\mathcal{P}$. For each rule $r = (b_1, ..., b_K \vdash h) \in \mathcal{P}$ and each predicate p appearing among r's premises, add a directed edge $(\mathrm{p}) \to (\mathrm{q})$ where q is h's predicate symbol. Then, label the edge $-$ if p is negated and $+$ otherwise. We say that G has a **negative cycle** if there exists a directed cycle that contains at least one negatively labeled edge. Logic programs that do not have negative cycles are called **stratified** (Abiteboul et al., 1995, §15.2). It is easy to see that any stratified logic program has a consistency-preserving fixpoint operator.

A.2 Forward Chaining

We give background details and pseudocode for the forward chaining algorithm; see Alg. 1. The algorithm takes a logic program $\mathcal{P} = \mathcal{R} \sqcup A$ with a set of ground goal theorems $\mathcal{H}_g$ and returns `T` (true) if all goal theorems in $\mathcal{H}_g$ can be proved. It uses a priority queue Q and a set $\mathcal{C}$. The priority queue Q is called the **agenda**. It keeps track of theorems that are to be used to prove new theorems. The order in which the axioms are pushed to Q will determine the order in which they are popped and Q may implement any arbitrary priority policy. For example, a last-in first-out (LIFO) policy yields a depth-first search (DFS) order, while a first-in first-out (FIFO) policy yields a breadth-first search (BFS) order. The set $\mathcal{C}$ is called a **chart**. The chart keeps track of theorems that have been proved. Theorems are added to $\mathcal{C}$ after they have been popped from Q.

When a theorem is popped, it is used to prove new theorems if applicable. The algorithm iterates over all rules in the program and proves all conclusions that can be proved from the popped premise together with theorems that have previously been added to $\mathcal{C}$. For a conclusion to be proved, there must exist an instantiation of the rule for which the premises in the instantiated ground rule are in $\mathcal{C}$. The algorithm iteratively removes theorems from $\mathcal{H}_g$ as they are proved, and terminates with the result `T` when $\mathcal{H}_g$ has been emptied. If all theorems that could be proved have been popped and $\mathcal{H}_g$ remains non-empty, the algorithm returns `F` (false). The algorithm may not terminate, however, if the minimal Herbrand model is infinitely large.

B Decidability of GSM Programs

Proposition 1 (Decidability)**.** *Let $\mathcal{P}_W = \mathcal{R}_W \sqcup A_W$ be a logic program where $\mathcal{R}_W$ is specified in Table 1 and A_W is a numerically consistent subset of $\mathcal{R}_W$'s extended Herbrand base $\overline{H}$. Then, inference in $\mathcal{P}_W$ is decidable, i.e., it is decidable to determine whether $h \in \mathrm{T}^*_{\mathcal{P}}(A_W)$ for an arbitrary $h \in \overline{H}$.*

Proof sketch. First, observe that the extended Herbrand base $\overline{H}$ of $\mathcal{P}_{\mathrm{W}}$ is countably infinite due to the unbounded sets of quantity constants $\mathbb{N}^q_{\boldsymbol{x}}$ and timestamp constants $\mathbb{N}^t_{\boldsymbol{x}}$. However, since the axioms A_{W} were chosen to be numerically consistent and the sets $\Sigma^{\mathrm{W}}_{\mathrm{p}}$, $2^A_{\boldsymbol{x}}$, and $E_{\boldsymbol{x}}$ are all finite, it follows that $\mathrm{T}^*_{\mathcal{P}}(A_{\mathrm{W}})$—or equivalently M—contains only finitely many theorems for each element of $\mathbb{N}^t_{\boldsymbol{x}}$. Consequently, reasoning in $\mathcal{P}_{\mathrm{W}}$ reduces to Presburger arithmetic, as the only built-in predicate that applies to timestamps is $\mathit{T}_{\mathbb{N}}{+}\mathit{1}$, present in Rule (2) in Table 1. Since Presburger arithmetic is famously decidable (Presburger, 1929; Haase, 2018), it follows that inference in $\mathcal{P}_{\mathrm{W}}$ is decidable. ■

C Data Generation

We adapt and apply Opedal et al.'s (2025) method for generating shortest proofs with axioms that are numerically consistent. App. C.2 introduces and discusses pseudocode (Alg. 2), App. C.3 explains how we use the algorithm to generate a proof along with irrelevant axioms, and App. C.4 shows that the algorithm indeed returns a *shortest* proof. To do so, we define a class of logic programs of which $\mathcal{P}_{\mathrm{W}}$ is a member, and we will present a generalization of Opedal et al.'s (2025) method

Algorithm 1 Generic forward chaining

function FORWARDCHAINING($\mathcal{P} = \mathcal{R} \sqcup A, \mathcal{H}_g$)
  ▷*program $\mathcal{P} = \mathcal{R} \sqcup A$, set of ground goal theorems $\mathcal{H}_g$*
  $Q \leftarrow \varnothing$ ▷*initialize agenda*
  $\mathcal{C} \leftarrow \varnothing$ ▷*initialize the chart / set of known atoms*
  for $b' \in A$: ▷*push axioms to agenda*
    Q.PUSH(b')
  while $Q \neq \varnothing$: ▷*pop premises while available*
    $b' \leftarrow Q$.POP() ▷*pop highest priority premise*
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{b'\}$ ▷*mark as proved by adding to chart*
    for $(b_1, ..., b_N \vdash h) \in \mathcal{R}$: ▷*iterate over all rules in program*
      if $\exists \theta \in \Theta(\mathcal{P}) : b_1/\theta, ..., b_N/\theta \vdash h/\theta$ **and** $b_1/\theta, ..., b_N/\theta \in \mathcal{C}$:
        ▷*there is a new inference rule available for which all premises have been previously proved*
        $h' \leftarrow h/\theta$ ▷*ground conclusion to the rule*
        if $h' \notin \mathcal{C}$: ▷*we proved a new theorem*
          Q.PUSH(h') ▷*push to agenda*
          if $h' \in \mathcal{H}_g$: ▷*we proved a new goal theorem*
            $\mathcal{H}_g \leftarrow \mathcal{H}_g \setminus \{h'\}$ ▷*mark as proved by removing from goal set*
            if $\mathcal{H}_g = \varnothing$: ▷*the goal set is empty so we have proved all goal theorems*
              return T
  ▷*all goal theorems could not be proved*
  return F

that generates shortest proofs under any program in this class. We start by defining this class of logic programs in App. C.1 below.

C.1 Additional Definitions

We define the **expanded Herbrand** base containing all ground and non-ground atoms—including negation (App. A.1)—as $\widetilde{H} \stackrel{\text{def}}{=} \{\mathrm{p}(t_1, ..., t_N) \mid \mathrm{p} \in \Sigma_\mathrm{p}, \mathsf{ar}(\mathrm{p}) = N, t_1, ..., t_N \in \Sigma_{\texttt{x}} \cup \Sigma_{\texttt{X}}\} \cup \{\neg\mathrm{p}(t_1, ..., t_N) \mid \mathrm{p} \in \Sigma_\mathrm{p}, \mathsf{ar}(\mathrm{p}) = N, t_1, ..., t_N \in \Sigma_{\texttt{x}} \cup \Sigma_{\texttt{X}}\}$. We let $C : \widetilde{H} \mapsto 2^{\Sigma_{\texttt{x}} \cup \Sigma_{\texttt{X}}}$ be a function that selects a set of terms from an input atom. We refer to $C(b)$ as the C**-terms** of b. One example of such a function in $\mathcal{R}_\mathrm{W}$ is to return the set of agent terms from a given atom, e.g., $C(\mathrm{cont}(\texttt{A}, \texttt{5}, \texttt{boat}, \texttt{1})) = \{\texttt{A}\}$ and $C(\mathrm{comp}(\texttt{A}, \texttt{\{haruki\}}, \texttt{2}, \texttt{boat}, \texttt{1})) = \{\texttt{A}, \texttt{\{haruki\}}\}$. In the following, we will also define C of an inference rule. Indeed, $r = b_1, ..., b_K \vdash h$, we define $C(r) \stackrel{\text{def}}{=} \bigcup_{k=1}^{K} C(b_k)$ as the union of the C-terms in the premises of the inference rule r. We similarly define C of a hyperedge: Given a hyperedge $e = \{b_1, ..., b_K\} \rightarrowtail h$, $C(e) \stackrel{\text{def}}{=} \bigcup_{k=1}^{K} C(b_k)$.

Definition 1. *We say a set of inference rules $\mathcal{R}$ is C-**conserving** if, for every inference rule $b_1, ..., b_K \vdash h$, we have $C(b_k) \neq \{\}$ for all k, $C(h) \neq \{\}$, and one of the following conditions holds:*

1. *The rule is an **introduction rule** for a predicate p, and can be written as*

$$b_1, ..., b_K \vdash \mathrm{p}(t_1, ..., t_N). \tag{2}$$

   Here, we require the premises to be non-p atoms. We require the C-terms in the conclusion appear in the premises: $C(\mathrm{p}(t_1, ..., t_N)) = \bigcup_{k=1}^{K} C(b_k)$. We also require that the C-terms of the premises are disjoint: $C(b_k) \cap C(b_j) = \varnothing$ for all $k \neq j$. We require that $\mathcal{R}$ have at most one introduction rule for each predicate.

2. *The rule is an **elimination rule** for a predicate p, and can be written as*

$$b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h. \tag{3}$$

   Here, we require that the C-term of the conclusion is distinct from that of the premises: $b_1, ..., b_{K-1}$, $C(h) \cap C(b_k) = \varnothing$ for all $k = 1, ..., K-1$. We also require that the C-term of the premise $\mathrm{p}(t_1, ..., t_N)$ contains the C-terms of all other premises and the conclusion: $C(h) \cup \bigcup_{k=1}^{K-1} C(b_k) \subseteq C(\mathrm{p}(t_1, ..., t_N))$. Finally, we require the C-terms of the other premises are disjoint: $C(b_k) \cap C(b_j) = \varnothing$ for all $0 < k, j < K$ and $k \neq j$.

3. *The rule is a **union rule**, where there is a built-in premise requiring the C-terms of the conclusion be equal to the union of the C-terms of the premises: $C(h) = \bigcup_{k=1}^{K} C(b_k)$. This rule must be unique in $\mathcal{R}$.*

4. *The rule is an **unused rule**: it contains premises that do not unify with the conclusion of any rule.*

Finally, if $\mathcal{R}$ has multiple elimination rules for a predicate p, we require that those rules be distinct in the following way: For any two such rules, written $b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h$ and $b'_1, ..., b'_{K-1}, \mathrm{p}(t'_1, ..., t'_N) \vdash h'$, for any θ such that $h/\theta = h'$, $\mathrm{p}(t_1, ..., t_N)/\theta \neq \mathrm{p}(t'_1, ..., t'_N)$.

Given a C-conserving set of inference rules $\mathcal{R}$, we will use the following short-hand notation to refer to specific subsets of rules:

- $\mathcal{R}_{\mathrm{I}}$ is the set of introduction rules in $\mathcal{R}$.
- $\mathcal{R}_{\mathrm{E}}$ is the set of elimination rules in $\mathcal{R}$.
- $\mathcal{R}_{\mathrm{U}}$ is the set of union rules in $\mathcal{R}$.

Definition 2. *Let $E\colon \widetilde{H} \mapsto 2^{\Sigma_{\boldsymbol{x}} \cup \Sigma_{\boldsymbol{\mathcal{I}}}}$ be a function. We say a set of inference rules $\mathcal{R}$ is E-**monotonic** if, for every inference rule $b_1, ..., b_K \vdash h$, we have $E(b_k) \neq \{\}$ for all k, $E(h) \neq \{\}$, and one of the following conditions holds:*

1. *The rule is an introduction rule; see above.*
2. *The rule is an elimination rule; see above.*
3. *The rule preserves E-terms: $E(b_1) = \cdots = E(b_K) = E(h)$.*

To prove the optimality of our generated proofs, we need additional constraints on the relationship between introduction and elimination rules. Let $C\colon \widetilde{H} \mapsto 2^{\Sigma_{\boldsymbol{x}} \cup \Sigma_{\boldsymbol{\mathcal{I}}}}$ be a function. Consider each elimination rule $b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h$, and let $C_b \stackrel{\text{def}}{=} \bigcup_{k=1}^{K-1} C(b_k)$ and $C_h \stackrel{\text{def}}{=} C(h)$. Construct a hypergraph using the following procedure: Start by mapping the atom $\mathrm{p}(t_1, ..., t_N)$ to a vertex v_0, i.e., we set $\ell(v_0) = (\mathrm{p}(t_1, ..., t_N))$. Repeat the following: For any vertex v that contains C-terms in both C_b and C_h (i.e., $C(\ell^{-1}(v)) \cap C_b \neq \varnothing$ and $C(\ell^{-1}(v)) \cap C_h \neq \varnothing$), and for any introduction rule $b'_1, ..., b'_K \vdash h' \in \mathcal{R}$, add a vertex for each unified premise b_k/θ where $\ell(h'/\theta) = v$ and a hyperedge from $\{\ell^{-1}(b_k/\theta)\}$ to $\ell^{-1}(v)$. We say the elimination rule is C-**locally reducible** if either: (1) the hypergraph contains no edges, or (2) there exists a vertex in this hypergraph that identifies with the conclusion of the elimination rule h. If all elimination rules in $\mathcal{R}$ are locally reducible, we say $\mathcal{R}$ is **in harmony**. One consequence of harmony is that, in any proof $\mathcal{P}$ containing a proof step with an elimination rule, if the subproof of the premise $\mathrm{p}(t_1, ..., t_N)$ does not contain any axioms with C-terms from both C_b and C_h, which is true of any proof generated by the procedure in App. C, then the conclusion of the elimination rule must be an axiom of the subproof. Therefore, $\mathcal{P}$ is not optimal.

We note that $\mathcal{R}_{\mathrm{W}}$ is **agent-conserving**, i.e., $\mathcal{R}_{\mathrm{W}}$ is C-conserving where C is the function that returns the agents of any atom. In particular, if we inspect the rules in Table 1, we observe that (1a) and (1b) are elimination rules for comp, and (1c) is the corresponding introduction rule. Rules (2a) and (2b) are elimination rules for transfer. Rule (4) is an elimination rule for rate. There are no corresponding introduction rules for transfer and rate. (5a) is an elimination rule for compeq and (5b) is the corresponding introduction rule. Rule (3) is a union rule. Rule (6) is an unused rule. $\mathcal{R}_{\mathrm{W}}$ is also **entity-monotonic** with (4) being the elimination rule for rate. $\mathcal{R}_{\mathrm{W}}$ is also in harmony, with respect to both agents and entities: (1a) and (1b) are each locally reducible with (1c). (5a) is locally reducible with a combination of (5b) and (1c).

A **ground** substitution $\{(\boldsymbol{\mathcal{I}}_m, t_m)\}_{m=1}^{M}$ is a typed substitution where $t_m \in \Sigma_{\boldsymbol{x}}^k$, for all $m \in [M]$. We define $\Theta_{\mathrm{G}}(\mathcal{P})$ to be the set of all ground substitutions under the logic program $\mathcal{P}$.

C.2 Applying Opedal et al.'s (2025) Method

Opedal et al. (2025) provide a method to generate the shortest proof for a goal h_g, i.e., the shortest $(\ell^{-1}(A_{\mathrm{W}}), \ell^{-1}(h_g))$-hyperpath that exists in a proof forest over the Herbrand base; see §3.2. We present the pseudocode in Alg. 2. The method takes any logic program $\mathcal{P}$ with C-conserving rules, e.g., $\mathcal{P}_{\mathrm{W}}$ as described in Table 1, a goal theorem h_g, a set of forbidden ground theorems $A_{\times}$, and a

Algorithm 2

function SAMPLESHORTESTPROOF($\mathcal{P}, h_g, A_\times, D$)
  ▷*logic program $\mathcal{P}$ (e.g., Table 1), goal theorem h_g, forbidden ground theorems $A_\times$, max depth D*
  let $\mathcal{R}$ be the inference rules in $\mathcal{P}$
  $\mathcal{P} \leftarrow \varepsilon$ ▷*initialize an empty hyperpath*
  QUEUE $\leftarrow [\,]$ ▷*initialize queue for conclusions to expand and their depth*
  QUEUE.PUSH($(0, h_g)$) ▷*h_g is distance 0 from h_g*
  SET $\leftarrow \{h_g\} \cup A_\times$
  while QUEUE $\neq \varnothing$:
    $(d, h) \leftarrow$ QUEUE.POP()
    if $d > D$: ▷*exit if stopping criterion has been met*
      break
    ▷*first, sample the inference rule for the next proof step*
    $\mathcal{R}' \leftarrow \{r \mid r = (b'_1, ..., b'_K \vdash h') \in \mathcal{R}_\mathrm{I} \cup \mathcal{R}_\mathrm{E} \cup \mathcal{R}_\mathrm{U}, \exists \theta \in \Theta_\mathrm{G}(\mathcal{P}) : h'/\theta = h\}$
    $r = (b'_1, ..., b'_K \vdash h') \sim$ SAMPLE($\mathcal{R}'$)
    ▷*next, we sample the substitution*
    $\Theta' \leftarrow \{\theta \mid \theta \in \Theta_\mathrm{G}(\mathcal{P}), h'/\theta = h, b'_k/\theta \notin$ SET$\}$
    ▷*sample novel and distinct C-terms whenever possible*
    let $C_\mathrm{SET} \stackrel{\text{def}}{=} \bigcup_{x \in \mathrm{SET}} C(x)$
    $\Theta' \leftarrow \Theta' \cap \{\theta \mid |\bigcup_{k=1}^{K} C(b'_k) \setminus C(h')| = |\bigcup_{k=1}^{K} C(b'_k/\theta) \setminus C_\mathrm{SET}|\}$
    if $r \in \mathcal{R}_\mathrm{U}$: ▷*sample widest possible union rules*
      $\Theta' \leftarrow \Theta' \cap \{\theta \mid C(b'_k/\theta) \cap C(b'_j/\theta) = \varnothing \text{ for } k \neq j\}$
    $\theta \sim$ SAMPLE(Θ')
    for $k = 1$ **up to** K:
      SET $\leftarrow$ SET $\cup \{b'_k/\theta\}$
      if $r \in \mathcal{R}_\mathrm{E}$ **and** $k \neq K$: ▷*the last premise of elimination rules are axioms*
        QUEUE.PUSH($(d + 1, b'_k/\theta)$)
    ▷*construct the hyperedge for this new proof step, and prepend it to the hyperpath*
    $\mathcal{P} \leftarrow (\{b_1, ..., b_K\} \rightarrowtail h) \circ \mathcal{P}$
  return $\mathcal{P}$

maximum depth D. The set $A_\times$ becomes relevant when generating irrelevant axioms, which will be discussed in App. C.3. For now, we assume that $A_\times = \varnothing$. The algorithm will terminate and return a proof of depth D. In our experiments with $\mathcal{P}_\mathrm{W}$, we use the following arguments: We sample the goal h_g to be a ground atom of the form cont(*A*, *Q*, *E*, *T*)$/\theta$ for some $\theta \in \Theta(\mathcal{P}_\mathrm{W})$. For the agent set we make an arbitrary choice, sampling an agent set with cardinality $\{1, 2, 3, 4\}$ with probabilities $\{1 \mapsto 1/2, 2 \mapsto 1/6, 3 \mapsto 1/6, 4 \mapsto 1/6\}$. We restrict the cardinality to 4 in order avoid generating GSM problems that are too large; see App. D.1 for dataset statistics. The timestamp is set to an arbitrary value larger than the value of D, which ensures that the timestamps remain in $\mathbb{N}^t_x$ throughout the generation procedure. We sample D uniformly at random from $\{1, 2, 3\}$ for every generated proof.

Sampling proceeds recursively in a top-down manner. We then sample an inference rule $b_1, ..., b_K \vdash h$, i.e., where h is the conclusion. This yields a tree. We then repeat the procedure recursively for each leaf node until the stopping criterion, i.e., required depth, has been reached. Importantly, all premises are sampled *without* replacement; this ensures that any individual theorem will not be used more than once in the generated proof, i.e., the proof is acyclic. For example, suppose we have cont(*{haruki}*, 5, *boat*, 1) and sample an instantiation of Rule (1a) in Table 1. We generate two new premises, cont(*{abu}*, 3, *boat*, 1) and comp(*{haruki}*, *{abu}*, 2, *boat*, 1), where the agent *{abu}* is a new agent that does not appear elsewhere in the proof. In Alg. 2, this is handled by maintaining a set SET of generated non-ground atoms, and by restricting the substitutions such that new C-terms are mapped to new objects. We only sample atoms that are not in SET.

We comment on two more details of Alg. 2. First, whenever we sample an elimination rule $b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h$, we require the last premise, $\mathrm{p}(t_1, ..., t_N)$, to be an axiom, i.e., we do not recursively apply the algorithm on this premise. This is due to the fact that the rules in $\mathcal{P}$ are in harmony, and so the addition of introduction rules preceding elimination rules could result in locally-reducible regions in the output proof, and so the proof would not be shortest. Second, if we

sample a union rule, we always require the C-terms of the premises to be singleton sets. Otherwise, a shorter proof could have been obtained by several instantiations of the union rule.[16]

Throughout the algorithm, we sample substitutions from $\Theta_{\text{G}}(\mathcal{P})$. By substituting under the constraints laid out by the arithmetic built-ins we ensure numerical consistency. However, numerical substitutions may fail due to unsatisfiable constraints, i.e., Θ' in Alg. 2 may become empty before we sample from it. We therefore perform rejection sampling until a successful substitution has been found. In a few cases, this may run prohibitively long. We therefore retry a maximum of 1000 times before rejecting the candidate proof and sampling a new one.

C.3 Generating Math Word Programs with Irrelevant Axioms

We apply Alg. 2 described in the previous subsection to generate irrelevant axioms. As mentioned in §4.3, we augment a logic program $\mathcal{P}_{\text{W}} = \mathcal{R}_{\text{W}} \sqcup A_{\text{W}}$—as sampled by the method described above—with M additional sets of irrelevant axioms $\widetilde{A}_1 \sqcup \cdots \sqcup \widetilde{A}_M$. In simple terms, we do so by applying the sampling procedure again, once for every $\widetilde{A}_m$, with additional restrictions to not generate duplicate axioms. As we will show later, this procedure for generating irrelevant axioms requires the additional constraint that the inference rules $\mathcal{R}$ in the logic program $\mathcal{P}$ be E-monotonic. With each application of the sampling procedure, we provide a perturbed goal argument h_g. At least one C-term or E-term of the original goal will be replaced with a new value (e.g., in $\mathcal{P}_{\text{W}}$, either the agent, the entity, or both will be perturbed). Precisely *which* term is perturbed depends on the experimental setting (see §4.3). The set of forbidden theorems $A_\times$ is initialized to be the theorems in the shortest proof (including the axioms) from $\mathcal{P}_{\text{W}} = \mathcal{R}_{\text{W}} \sqcup A_{\text{W}}$, thus, avoiding generating those theorems. The same is done incrementally for the irrelevant axioms as they are generated, guaranteeing disjoint sets.

Moreover, for practical reasons, the distribution from which we sample is slightly modified. In particular, we restrict cont predicates to have singleton agent sets; otherwise, the number of irrelevant axioms could get very large. Furthermore, we exclude the rate predicate, as we found that overlapping agents and entities could sometimes be hard to instantiate with a rate atom, since it requires a particular semantic relationship between the two entity terms in the atom, e.g., *"apples"* per *"basket"*. Enforcing these restrictions led to substantially more efficient rejection sampling during the substitution part of Alg. 2.

When ordering the axioms in natural language, it is undesirable to have a predictable ordering of relevant and irrelevant axioms, e.g., appending all irrelevant axioms to follow the relevant ones. We therefore randomly reorder them in a manner that respects the ordering of the values of the timestamp.

C.4 Uniqueness and Optimality of Generated Proofs

Here, we will show that the above procedure for generating proofs produces a shortest proof—a proof from axioms A to a goal h_g for which there does not exist a shorter proof of h_g from A. Furthermore, we will show that each generated proof is *unique*: There are no other shortest proofs with the same axioms and goal theorem.

C-conservation, monotonicity, and harmony of the inference rules alone is insufficient to guarantee that the proofs generated by the procedure described in App. C are shortest. We need additional properties of the generated proofs, as well as a few additional definitions.

A **linear chain** is a sequence of hyperedges $e_1, ..., e_M$ where the head of every edge is in the tail of the next edge (except for the last edge). More precisely, for all m, we have $e_m = \{b_{m,1}, ..., b_{m,K}\} \rightarrowtail h_m$, and for all $0 < m < M$, $h_m \in \{b_{m+1,1}, b_{m+1,2}, ...\}$.

Given an (A, h_g)-proof $\mathcal{P} = ((V, E), \ell)$, and an atom $x \in \overline{H}$ in the extended Herbrand base $\overline{H}$ (App. A.1) where $\ell^{-1}(x) \in V$, a **subproof** of x is an (A, x)-proof $\mathcal{P}_x = ((V_x, E_x), \ell)$ where $V_x \subseteq V$ and $E_x \subseteq E$. Using this, we can define the subproof relation: If $\mathcal{P}_x$ is a subproof of some atom in $\mathcal{P}$, we write $\mathcal{P}_x \preceq \mathcal{P}$. We say a subproof $\mathcal{P}_x$ of x *contains* an atom y, or similarly, y *is in* $\mathcal{P}_x$, if there is an (A, x)-hyperpath where $\ell^{-1}(y)$ is in the head or tail of any edge in the hyperpath. Similarly, we say a subproof $\mathcal{P}_x$ of x *contains* a C-term t, or t *is in* $\mathcal{P}_x$, if there is some y such that $\mathcal{P}_x$ contains y and $t \in C(y)$.

[16] This is analogous to folding and unfolding (Tamaki & Sato, 1984).

Theorem 1 (Generated proofs are shortest and unique)**.** *Let $\mathcal{R}$ be a set of inference rules that are C-conserving, E-monotonic, and in harmony,*[17] *let $\mathcal{P}$ be an (A, h_g)-proof in $\mathcal{R} \sqcup A$ generated by the procedure in App. C.3, and let $\widetilde{A}$ be the set of irrelevant axioms generated by App. C.3. There does* not *exist an (A', h_g)-proof $\mathcal{P}'$ in $\mathcal{R} \sqcup A \sqcup \widetilde{A}$ such that $|\mathcal{P}'| \leq |\mathcal{P}|$.*

To prove Thm. 1 we introduce two lemmas. First, in Lemma 1, we show that the shortest proofs (without irrelevant axioms) are unique. That is, for a given goal theorem and set of axioms, there is no shortest proof that is different than the one we generate. Then, we show that the irrelevant axioms generated in App. C.3 generated through Alg. 2 can not be used to yield another shortest proof of the goal theorem; this is done in Lemma 2. Taken together, these lemmas imply Thm. 1.

Lemma 1. *Let $\mathcal{R}$ be a C-conserving set of inference rules, and let A be a set of axioms. Then, for all theorems h_g in $\mathcal{R} \sqcup A$ such that (A, h_g)-proof $\mathcal{P}$ could be generated by Alg. 2, we have that $\mathcal{P}$ is h_g's unique shortest proof in $\mathcal{R} \sqcup A$.*

Proof. We perform structural induction on the subproof relation $\preceq$.

We first observe that $\mathcal{P}$ can not have cycles: By definition, a cycle contains several vertices labeled with the same theorem. This is not possible since Alg. 2 maintains a set of already sampled theorems SET that can never be sampled again: Specifically, line 16 excludes theorems in SET from being sampled and line 24 adds generated theorems to SET.

Base Case. The base case is the smallest possible proof containing a single axiom that is also the goal theorem, i.e., an (A, h_g)-proof $\mathcal{P}$ of h_g with $|\mathcal{P}| = 1$. This is clearly the unique shortest proof in this case.

Inductive Case. Consider the last hyperedge $\{\ell^{-1}(b_1), ..., \ell^{-1}(b_K)\} \rightarrowtail \ell^{-1}(h)$, i.e., the hyperedge for which the head is labeled with the goal theorem h of the subproof. The inductive hypothesis states that for any atom $x \neq h$ in $\mathcal{P}$, the subproof of x in $\mathcal{P}$ is the unique shortest (A, x)-proof in $\mathcal{R} \sqcup A$. We consider the various cases of inference rules in the logic program for the last hyperedge and show that the unique shortest proof of the conclusion is obtained by combining the unique shortest proofs of the premises under the inference rule corresponding to that hyperedge.

We now consider four cases.

Case 1: (Introduction). The inference rule corresponding to the last hyperedge is an instantiation of an introduction rule:

$$b_1, ..., b_K \vdash \mathrm{p}(t_1, ..., t_N). \tag{4}$$

Because there are no other introduction rules for p, $C(\mathrm{p}(t_1, ..., t_N)) = \bigcup_{k=1}^{K} C(b_k)$, and the atoms in the subproofs of the premises are disjoint, the above introduction rule is the only way to prove $\mathrm{p}(t_1, ..., t_N)$. Thus, we conclude that $\mathcal{P}$ is the unique shortest proof of $\mathrm{p}(t_1, ..., t_N)$.

Case 2: (Elimination). The inference rule corresponding to the last hyperedge is an instantiation of an elimination rule: $b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h$. Due to the restriction in the generative process—see App. C.2—the premise $\mathrm{p}(t_1, ..., t_N)$ is not expanded when generating $\mathcal{P}$ because it is an axiom. Furthermore, since $C(h) \cap C(b_k) = \varnothing$ for all premises b_k, it is impossible to prove h using only the atoms in the subproofs of the premises $b_1, ..., b_{K-1}$. In fact, since $C(h) \subseteq C(\mathrm{p}(t_1, ..., t_N))$, the axiom $\mathrm{p}(t_1, ..., t_N)$ *must* appear in any proof of h. The logic program may have other elimination rules for the same predicate p, but since such rules are distinct due to C-conservation, those other rules cannot be used to derive h from $b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N)$. Therefore, we conclude that $\mathcal{P}$ is the unique shortest proof of h.

Case 3: (Union). The inference rule corresponding to the last hyperedge is an instantiation of a union rule: $b_1, ..., b_K \vdash h$. Since the generation procedure in App. C.2 only generates hyperedges corresponding to union rules where the premises have disjoint C-terms, i.e., $C(b_k) \cap C(b_j) = \varnothing$ for all $k \neq j$, and there is no way to prove h other than the union rule, this case is identical to *Case 1* above.

Case 4: (Unused). The inference rule corresponding to the last hyperedge is an instantiation of an unused rule. But the procedure in App. C.2 never includes such inference rules, so this is impossible.

■

[17] With respect to both C and E.

Lemma 2. *Let $\mathcal{R}$ be a set of inference rules that is C-conserving, E-monotonic, and in harmony, let (A, h_g)-proof $\mathcal{P}$ be the proof in $\mathcal{R} \sqcup A$ generated by App. C.3, and let $\widetilde{A}$ be the set of irrelevant axioms generated by App. C.3. Then, there does* not *exist an (A', h_g)-proof $\mathcal{P}'$ in $\mathcal{R} \sqcup A \sqcup \widetilde{A}$ such that $A' \neq A$ and $|\mathcal{P}'| \leq |\mathcal{P}|$.*

Proof. We offer a proof by contradiction. By way of contradiction, assume there exists an (A', h_g)-proof $\mathcal{P}'$ in $\mathcal{R} \sqcup A \sqcup \widetilde{A}$ such that $A' \neq A$ and $|\mathcal{P}'| \leq |\mathcal{P}|$. Take $\mathcal{P}'$ to be the shortest such proof. Because, by construction of the algorithm, the proof $\mathcal{P}$ consists of a single hyperpath starting from A. Thus, $A' \neq A$ implies there exists an $a \in A'$ where $a \notin A$. Recall that $\mathcal{P} = ((V, E), \ell)$ is itself a proof forest. Let $C(\mathcal{P}) \stackrel{\text{def}}{=} \bigcup_{e \in E} C(e)$ be the set of C-terms in $\mathcal{P}$, i.e., the **relevant C-terms**. Similarly, let $E(\mathcal{P}) \stackrel{\text{def}}{=} \bigcup_{e \in E} E(e)$ be set of E-terms in $\mathcal{P}$, i.e., the relevant E-terms. We now consider two cases.

Case 1: $C(a) \subseteq C(\mathcal{P})$ (The irrelevant axiom only has relevant C-terms). The generation procedure described in App. C.3 can only generate an irrelevant atom with relevant C-terms when generating proof trees that have E-terms that are distinct from those in the relevant proof. Therefore, the E-terms in $\mathcal{P}'$, i.e., the irrelevant E-terms, are disjoint from those in $\mathcal{P}$, i.e., the relevant E-terms. Take the linear chain $e_1, ..., e_M \in \mathcal{P}'$ such that a is in the tail of e_1 and h_g is the head of e_M. Let h_m be the head of edge e_m. Select the first e_m such that $E(h_m)$ contains an E-term that is not relevant and $E(h_{m+1})$ has only relevant E-terms. There must be at least one such edge since h_g has only relevant E-terms but a does not. Because $\mathcal{R}$ is E-monotonic, only elimination rules allow an E-term from a premise to be absent from the conclusion, and so e_{m+1} must be an instantiation of an elimination rule

$$b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h, \tag{5}$$

and $E(h) \cup E(b_k) = E(\mathrm{p}(t_1, ..., t_N))$, for any k. The E-terms of h_m and h_{m+1} are included in $E(\mathrm{p}(t_1, ..., t_N))$. Thus, the premise $\mathrm{p}(t_1, ..., t_N)$ must contain both relevant and irrelevant E-terms. Note that in the generation procedure described in App. C.3, no axiom is generated that contains both relevant and irrelevant E-terms. Therefore, $\mathrm{p}(t_1, ..., t_N)$ must be derived from axioms other than a. However, because $\mathcal{R}$ is in harmony, the subproof of this premise must contain h, and so the proof $\mathcal{P}'$ is not shortest, which is a contradiction.

Case 2 $C(a) \nsubseteq C(\mathcal{P})$ (The irrelevant axiom has an irrelevant C-term). Take the linear chain $e_1, ..., e_M \in \mathcal{P}'$ such that a is in the tail of e_1 and h_g is the head of e_M. Let h_m be the head of edge e_m. Select the first e_m such that $C(h_m) \nsubseteq C(\mathcal{P})$ and $C(h_{m+1}) \subseteq C(\mathcal{P})$. There must be at least one such edge since h_g has only relevant C-terms but a does not. Since the $\mathcal{R}$ is C-conserving, only elimination rules allow a C-term from a premise to be absent from the conclusion, and so e_{m+1} must be an instantiation of an elimination rule

$$b_1, ..., b_{K-1}, \mathrm{p}(t_1, ..., t_N) \vdash h, \tag{6}$$

and $C(h) \cup \bigcup_{j=1}^{K-1} C(b_j) \subseteq C(\mathrm{p}(t_1, ..., t_N))$. That is, the C-terms of h_m and h_{m+1} are included in $C(\mathrm{p}(t_1, ..., t_N))$. Thus, the premise $\mathrm{p}(t_1, ..., t_N)$ must contain both relevant and irrelevant C-terms. Note that in the generation procedure described in App. C.3, no axiom is generated that contains both relevant and irrelevant C-terms. Therefore, $\mathrm{p}(t_1, ..., t_N)$ must be derived from other axioms. However, since $\mathcal{R}$ is in harmony, the subproof of this premise must contain h, and so the proof $\mathcal{P}'$ is not shortest, which is a contradiction. ■

We can now prove Thm. 1 using the above lemmas.

Theorem 1 (Generated proofs are shortest and unique)**.** *Let $\mathcal{R}$ be a set of inference rules that are C-conserving, E-monotonic, and in harmony,*[18] *let $\mathcal{P}$ be an (A, h_g)-proof in $\mathcal{R} \sqcup A$ generated by the procedure in App. C.3, and let $\widetilde{A}$ be the set of irrelevant axioms generated by App. C.3. There does* not *exist an (A', h_g)-proof $\mathcal{P}'$ in $\mathcal{R} \sqcup A \sqcup \widetilde{A}$ such that $|\mathcal{P}'| \leq |\mathcal{P}|$.*

Proof. By Lemma 1, there exists one unique shortest proof $\mathcal{P}$ of a goal theorem from a set of relevant axioms and this is the proof we generate as the ground-truth proof. By Lemma 2, there will be no additional proofs $\mathcal{P}'$ where $|\mathcal{P}'| \leq |\mathcal{P}|$ when generating irrelevant axioms through our procedure. Therefore, all our generated proof systems have a unique shortest proof, which is identical to the ground-truth proof that is generated through our procedure. ■

[18] With respect to both C and E.

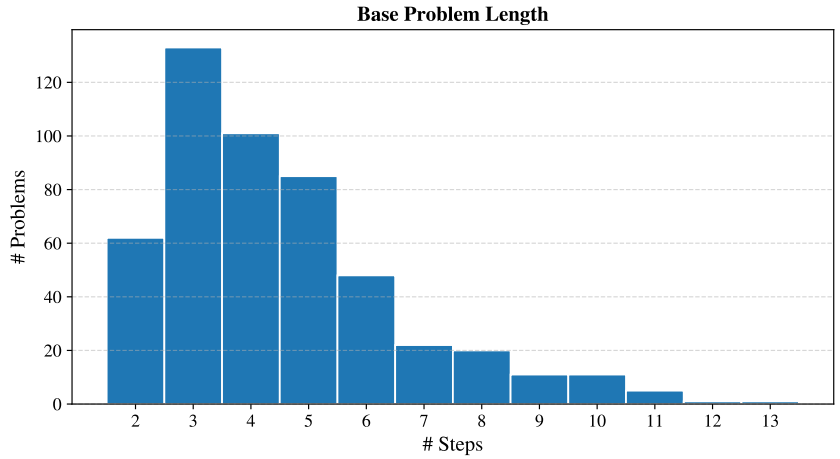

Figure 4: Histogram of the number of hyperedges (i.e., inference rule applications) present in the shortest proof for the GSM programs in our dataset.

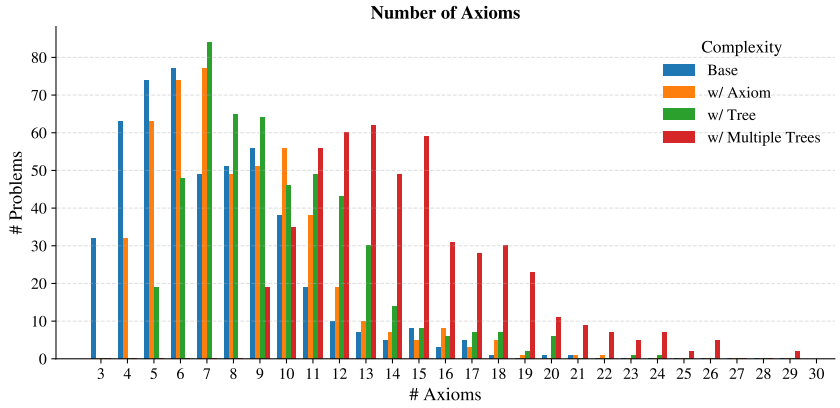

Figure 5: Histogram of the total number of axioms, including irrelevant ones, per type of complexity of the GSM programs in our dataset.

D More Details on Experiments

D.1 Dataset Statistics

Here, we present more statistics on the datasets used in our experiments (§5). Fig. 4 shows the distribution of the number of edges in the proofs in the base problems, which is equivalent to the number of non-axiom theorems. The numbers range from 2 to 13.[19] The distribution is skewed towards shorter problems. This was done by design in order to prevent saturation when evaluating models with different levels of capabilities.

Fig. 5 shows the number of axioms the GSM problems contain, including irrelevant ones. As expected, adding one irrelevant axiom (w/ axiom) yields a distribution that is shifted by exactly one additional axiom as compared to the base distribution. When adding a single irrelevant tree (w/ tree), the distribution shifts further and spreads out. This trend is further amplified when adding multiple irrelevant trees (w/ multiple trees).

D.2 Prompt

See Fig. 6 for the prompt used in our experiments.

[19]For reference, GSM8K (Cobbe et al., 2021) includes problems with 2 to 8 steps.

[PROMPT 1]

You are a helpful assistant tasked with solving math word problems. You follow the formatting of the problems and the solutions as given in the examples below.

Q: Lib possesses 20 puzzles. Shanna has 4 puzzles more than Lib. Shanna then donates 5 puzzles to Costa. Shanna has 6 puzzles more than Valeria. Valeria has 11 puzzles more than Clementius. How many puzzles does Clementius have totaled?

A: Let's think step by step.
1. Lib possesses 20 puzzles.
2. Shanna has 4 puzzles more than Lib.
3. So Shanna has 20 + 4 = 24 puzzles.
4. Shanna then donates 5 puzzles to Costa.
5. So Shanna has 24 - 5 = 19 puzzles.
6. Shanna has 6 puzzles more than Valeria.
7. So Valeria has 19 - 6 = 13 puzzles.
8. Valeria has 11 puzzles more than Clementius.
9. So Clementius has 13 - 11 = 2 puzzles.
Therefore, the answer (arabic numerals) is 2.

[...repeat...]

Q: Stacee owns 18 necklaces. Stacee then gets 14 more necklaces from Felic. Valencia has 11 necklaces less than Stacee. How many necklaces does Valencia have?

A: Let's think step by step.

[GENERATION 1]

1. Stacee owns 18 necklaces.
2. Stacee then gets 14 more necklaces from Felic.
3. So Stacee has 18 + 14 = 32 necklaces.
4. Valencia has 11 necklaces less than Stacee.
5. So Valencia has 32 - 11 = 21 necklaces.

[PROMPT 2]

Therefore, the answer (arabic numerals) is

[GENERATION 2]

21

Figure 6: Two-stage prompting method used in our experiments. We first prompt the model with in-context examples (5-shot) and the test problem. After generation, we re-prompt the model to generate the final numerical answer.

D.3 Example Proof

In this section we give an example proof in natural language. Consider the following problem, with the irrelevant axioms marked in **bold**:

> **Anatola has 67 keys. Ichabod has 23 packets. Clayborne has 55 crayons.** Pieter has 20 parrots. **Ichabod has 30 packets less than Winnah.** Pieter then gives Yorgos 6 parrots. **Clayborne has 96 crayons less than Imogene. Anatola has 55 keys less than Rahel.** Dennie has 19 parrots. Dianna owns 4 parrots. The number of parrots that Edgar has more than Pieter is the same as the difference between the number of parrots that Dennie has compared to Dianna. Edgar has 3 parrots more than Eddie. How many parrots does Eddie have in all?

Note that the only axioms that are relevant are the ones containing information about parrots. The shortest proof for this problem requires the application of four inference rules and can be written in natural language as follows:

1. Pieter has 20 parrots.
2. Pieter then gives Yorgos 6 parrots.
3. So Pieter has 20 - 6 = 14 parrots.
4. Dennie has 19 parrots.
5. Dianna owns 4 parrots.

6. So the difference between the number of parrots Dennie and Dianna have is 19 - 4 = 15.
7. The number of parrots that Edgar has more than Pieter is the same as the difference between the number of parrots that Dennie has compared to Dianna.
8. So Edgar has 14 + 15 = 29 parrots.
9. Edgar has 3 parrots more than Eddie.
10. So Eddie has 29 - 3 = 26 parrots.

In this case, employing a best-first search heuristic based on lexical overlap with the query (i.e., prioritizing theorems that contain information about parrots) will yield the above shortest proof. Llama-3.1-8B-Instruct did not opt for that strategy. Instead, it generated the following, inefficient solution to this problem:

1. Anatola has 67 keys.
2. Anatola has 55 keys less than Rahel.
3. So Rahel has 67 + 55 = 122 keys.
4. Ichabod has 30 packets less than Winnah.
5. So Winnah has 23 + 30 = 53 packets.
6. Clayborne has 96 crayons less than Imogene.
7. So Imogene has 55 + 96 = 151 crayons.
8. Pieter has 20 parrots.
9. Pieter then gives Yorgos 6 parrots.
10. So Pieter has 20 - 6 = 14 parrots.
11. The number of parrots that Edgar has more than Pieter is the same as the difference between the number of parrots that Dennie has compared to Dianna.
12. Dennie has 19 parrots and Dianna has 4 parrots.
13. The difference between Dennie and Dianna is 19 - 4 = 15.
14. So Edgar has 15 more parrots than Pieter.
15. Edgar has 15 + 14 = 29 parrots.
16. Edgar has 3 parrots more than Eddie.
17. So Eddie has 29 - 3 = 26 parrots.

Therefore, the answer (arabic numerals) is 26.

D.4 Further Empirical Results

Accuracy Stratified by Agent and Entity Overlap. Fig. 7 shows further stratified results that were mentioned in §5.1. We briefly note that these results suggest that the models might make use of a heuristic based on such overlap. (Recall that the problem sets are otherwise identical.) One could reasonably expect that a model could learn such a heuristic, based on previous work suggesting that models may derive the conclusion to a proof step greedily within a forward pass before initiating the step in the output tokens (Kudo et al., 2024; Wu et al., 2024).

Parsing Reasoning Model Outputs. Since QwQ-32B and DeepSeek-R1 did not generate output corresponding to strings in our verbalized program, we performed a separate, more crude analysis for those two models. Rather than matching the full generated token sequence, we match only the arithmetic expressions. They are matched with the ground-truth expressions from the built-ins of the corresponding inference rules. This can only be done for theorems that are not axioms, since only those require an arithmetic built-in (i.e, arithmetic expression) to prove. Arithmetic expressions are extracted flexibly, accounting for formatting irregularities and natural language that may be interleaved. The parser may also predict no match for a particular model output if no matches are found.

We manually verified parsing accuracy (macro-average) on model outputs over 20 randomly selected example problems. The parser predicted the correct match (or correctly predicted no match) in 94.4% of cases for QwQ-32B and 94.8% of cases for DeepSeek-R1.

Table 4 displays the results. We observe results that are overall consistent with the main conclusions of the paper; even state-of-the-art models like DeepSeek-R1—despite performing well in terms of accuracy (Table 2)—appear to generate theorems that are irrelevant to the goal theorem. As with the two models discussed in the main text, the efficiency scores decrease when there is agent and/or entity overlap between the irrelevant axioms and the goal theorem.

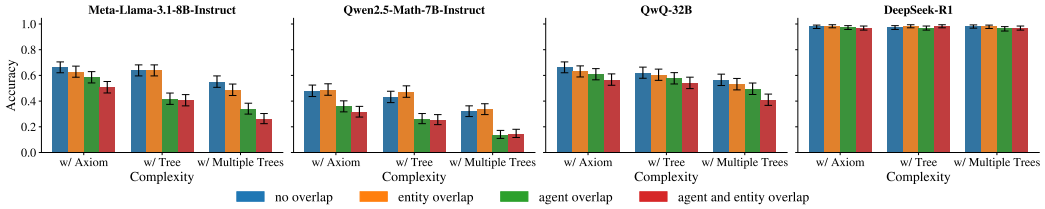

Figure 7: Results on problems with irrelevant axioms from Table 2 as average answer accuracy (%) when stratified by different kinds of lexical overlap between the irrelevant axioms and the query. The error bars show the 95% confidence interval using Wilson (1927) score intervals.

		Non-ground Queries			
		None	Entity	Agent	Both
QwQ-32B	Efficiency (non-axioms)	77.1	63.8	72.2	63.1
DeepSeek-R1	Efficiency (non-axioms)	89.4	67.7	77.3	67.8

Table 4: Additional efficiency analysis for QwQ-32B and DeepSeek-R1, using a more crude parser that only extracts arithmetic expressions corresponding to the arithmetic built-ins in Table 1. The efficiency score can therefore only be presented for theorems that are not axioms. Results are stratified and computed in the same way as §5.2. The differences in efficiency scores across datasets of different lexical overlaps are significant ($p < 0.001$) according to one-way ANOVA analyses.

Search Order. While our results in §5.2 suggest that LMs make use of information in the query to prove goals, they also reveal that LMs generate irrelevant theorems. Thus, LMs appear to use some heuristic, albeit an imperfect one, in their reasoning. Here, we present a limited analysis on whether this heuristic could be in combination with either DFS or BFS. We do so by examining the order in which intermediate theorems, both relevant and irrelevant, are generated by the LM, and comparing this order to the respective reference orders. Furthermore, we compare the LM's ordering to the theorems in the shortest proof as visited in DFS order as a control.

Before presenting the results, we make note of a few limitations of this analysis. First, the only irrelevant theorems we consider in the DFS and BFS orderings are those that are generated by Alg. 2 when sampling the axioms for the irrelevant goal theorems $\widetilde{h}_1 ... \widetilde{h}_M$ (§4.3). In addition, we note that the in-context examples were ordered according to DFS; see §5. However, the examples were chosen so that the ordering of theorems that are not axioms would be the same under both DFS and BFS. We therefore ignore the axioms as we compare the orderings.

Concretely, the DFS and BFS orderings correspond to the order in which atoms are popped from the chart $\mathcal{C}$ in Alg. 1, where: (i) $\mathcal{C}$ is implemented as a stack for DFS and a queue for BFS, (ii) the axioms are popped from the agenda Q in the order in which they are presented (i.e., pushed) to the model in natural language (as described in §5), and (iii) we do not include pops of axioms.

As our metric, we compute the Levenshtein distance between the sequence of indices produced by the LM and the ground truth reference ordering under the different search orders, normalized by the longest of the two sequences. The results are shown in Fig. 8. We observe that the models' search orders are closer to DFS than to BFS across all types of query overlap. Furthermore, for Llama-3.1, they are closer to the DFS-based ordering required for the shortest proof than DFS when there is no agent or entity overlap between the goal and the irrelevant axioms. Qwen2.5-Math appears to be less efficient however, consistent with the results in §5.2. These findings suggest that LMs tend to follow a depth-first exploration of the proof space in combination with the superficial heuristic. However, future work should perform a more rigorous analysis to confirm these preliminary findings.

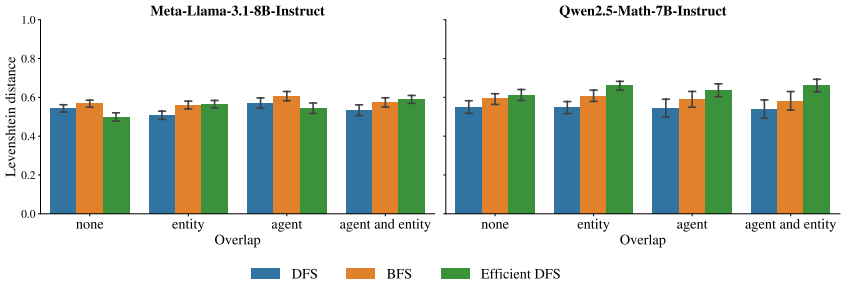

Figure 8: Average Levenshtein distance between Llama-3.1-8B-Instruct's search order and depth-first search (DFS), breadth-first search (BFS), and an efficient DFS which only visits relevant theorems. The plot is based on problems for which the model gave the correct answer. The Levenshtein distances are normalized by the length of the longest string to take values in $[0, 1]$; lower values mean shorter distances.