# SAGE: Satisfiability Guided Decoding for Mathematical Reasoning

**Tamanna Hossain and Sameer Singh and Kareem Ahmed**
University of California, Irvine
{tthossai, sameer}@ics.uci.edu, kareem.yousrii@gmail.com

## Abstract

Chain-of-thought prompting improves mathematical reasoning, but generated reasoning chains can appear locally plausible while being globally inconsistent. Later steps may derive equations inconsistent with claims asserted earlier. We propose SAGE, a training-free decoding approach that treats symbolic consistency as probabilistic conditioning over mathematical claims, conditioning each new claim on the symbolic state accumulated from earlier steps. SAGE approximates this conditional distribution online by resampling at mathematical span boundaries: candidate continuations are parsed into formal claims and selected according to satisfiability with the accumulated symbolic state. This statement-level decoding avoids intractable token-level constrained decoding while still using solver feedback online, before a complete solution has been generated. Across MATH-500, Gaokao, and Minerva, SAGE improves LLaMA, DeepSeek, and Qwen models over temperature sampling. On MATH-500, it improves accuracy by up to 6.3% and by 3.1% on average across four models. On Gaokao and Minerva datasets, SAGE yields consistent gains across models, improving Gaokao and Minerva `pass@3` by up to 2% and 3%, respectively. We release our code at https://github.com/tamannahossainkay/sage

## 1 Introduction

Chain-of-thought prompting has been pivotal to improving the performance of language models (LMs) on mathematical and symbolic reasoning tasks by eliciting intermediate reasoning steps (Wei et al., 2022; Kojima et al., 2022). However, producing more reasoning tokens also creates more opportunities for symbolic inconsistency: a reasoning trace may derive one value for a quantity, only to later assign it a different value, or perform an algebraic simplification contradicting constraints established earlier in the solution. This observation suggests a formal lens. If the mathematical claims in a reasoning trace are extracted as constraints, then one basic requirement is *satisfiability*: there should exist an assignment under which all accepted claims are simultaneously true.

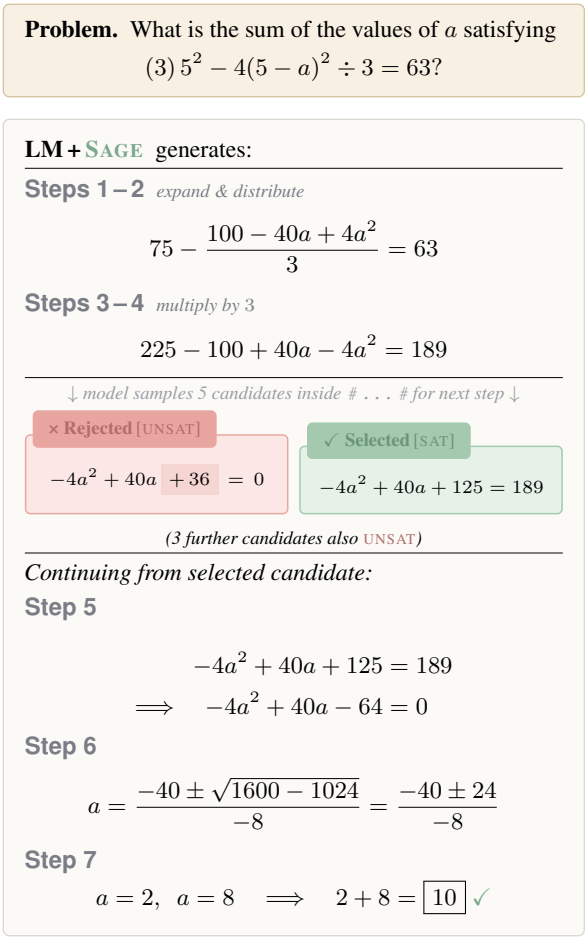



Figure 1: SAGE **performs span constrained resampling**. At a math span, the LM proposes several candidate simplifications; each is checked against the symbolic state induced by earlier accepted steps. Here, the rejected candidate is UNSAT as it simplifies $225 - 100 - 189$ to 36 instead of $-64$, while the selected candidate preserves satisfiability, leading to the correct answer.

Prior work has proposed to augment satisfiabil-



*Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 49649–49665
October 24-29, 2026 ©2026 Association for Computational Linguistics

ity with mathematical reasoning in LMs. However, this is typically carried out in a post-hoc manner, with the solver outside the decoding loop: they either use satisfiability to filter or rank complete Best-of-$N$ reasoning traces (Zhou et al., 2024), or delegate reasoning to a solver after the LM translates the problem into declarative constraints (Ye et al., 2023). While such approaches demonstrate the value of satisfiability as a reasoning signal, they only apply this signal after a full solution or full formalization has already been generated. In contrast, incorporating satisfiability during decoding allows inconsistencies to be detected and corrected locally at mathematical spans, avoiding the need to repeatedly generate and score complete traces.

We introduce *Satisfiability-Guided Decoding* (SAGE), a training-free, inference-time approach for guiding LM reasoning traces toward *symbolic self-consistency*. The core idea underlying SAGE is *statement-level constrained resampling*: rather than constraining the next-token distribution directly, SAGE constrains the distribution over mathematical statements. At each mathematical span boundary, the LM proposes candidate continuations from its local span distribution; a lightweight parser maps each parseable span to formal claims; and a solver checks whether adding those claims to the accumulated symbolic state preserves satisfiability. Generation resumes from a satisfiable candidate. Figure 1 gives an approach overview.

SAGE uses satisfiability as an *online* decoding signal rather than as a post-hoc filter over completed solutions (Figure 2). Our intuition is that even when the LM's preferred next mathematical claim is inconsistent, nearby samples from the same local span distribution may remain compatible with the trace constructed so far. SAGE therefore does not replace LM reasoning with solver-based reasoning; it locally resamples the model's immediate mathematical claims subject to consistency with the accumulated symbolic state. Unlike approaches that autoformalize an entire solution into a theorem-prover input, SAGE extracts constraints directly from intermediate statements and checks them incrementally during generation. This yields a lightweight, model-agnostic method for improving symbolic self-consistency at inference time, without fine-tuning or a trained verifier.

We evaluate SAGE across MATH-500, Gaokao, and Minerva, finding consistent gains for LLaMA and DeepSeek models over temperature sampling without any fine-tuning or trained verifier (§4). On MATH-500, SAGE improves accuracy by up to 6.3 percentage points and by 3.1 points on average across four models. Gains persist across subject areas and difficulty levels, with the largest improvements concentrated at intermediate difficulty levels. On the harder Gaokao and Minerva benchmarks, SAGE improves `pass@3` by up to 2 and 3 points.

In a compute-matched comparison, SAGE outperforms majority voting over three complete solutions (MV@3) by 3.1 percentage points overall, even though MV@3 is given a larger $3\times$ token-generation budget compared to SAGE's approximately $2\times$ overhead. Gains are observed in every dataset–model setting, with non-overlapping mean $\pm$ standard-deviation intervals in 7 of 12 settings. Diagnostics of the last-fork solver status further show that correct solutions have more SAT outcomes, while incorrect solutions have more UNSAT outcomes, indicating that solver verification provides a meaningful correctness signal.

Together, these results show that mathematical reasoning improves when generation is constrained to remain globally consistent. SAGE enforces this consistency online, checking each new mathematical claim against the symbolic state built from earlier steps and steering the model away from continuations that contradict its own reasoning.

## 2 Preliminaries and Notation

### 2.1 A Language Model Induces a Distribution Over a Problem's Reasoning Traces

Let $\mathbf{s}$ denote a problem statement and let $\mathbf{y}_{1:T} \coloneqq [\mathrm{y}_1 \mathrm{y}_2 \dots \mathrm{y}_T]$ denote a reasoning trace of arbitrary length $T$, where $\mathrm{y}_i$ is the instantiation of random variable $Y_i$ over tokens at time $i$ and takes values from a vocabulary $\mathbb{V} = \{1, \dots, V\}$. An autoregressive LM $p_\theta$ defines a distribution over traces

$$p_\theta(\mathbf{y}_{1:T} \mid \mathbf{s}) = \prod_{t=1}^{T} p_\theta(\mathrm{y}_t \mid \mathbf{s}, \mathbf{y}_{<t}). \qquad (1)$$

In standard decoding, one samples, greedily or stochastically, tokens w.r.t. the next-token conditionals until the `<EOS>` token is encountered.

In this work, we are interested in *reasoning traces* containing explicit mathematical statements. We prompt the model to delimit such statements using `#...#`. This induces a natural decomposition of the trace into ordinary text and mathematical spans. Let $\mathbf{x}_{1:t}$ be the textual prefix preceding a mathematical span, then the LM also induces a

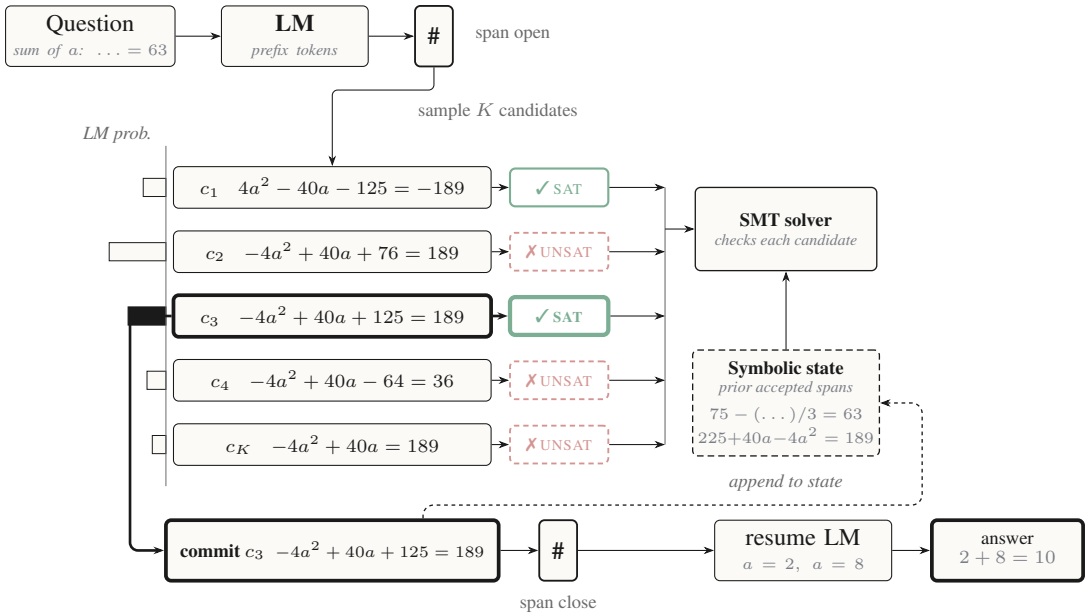



Figure 2: SAGE **intercepts at math-span boundaries (#) to enforce arithmetic consistency during generation**. Inside a span, $K$ complete candidate continuations are sampled from the LM's distribution (bars at left), and each is checked by an SMT solver against the symbolic state accumulated from prior accepted spans. The solver partitions candidates by satisfiability: here $c_1$ and $c_3$ both pass, while $c_2$, the LM's highest-probability continuation, is UNSAT and would have introduced an error. SAGE commits a satisfiable candidate ($c_3$) using the LM's own preference among them, appends its equation to the symbolic state for downstream spans, and resumes generation. SAGE recovers the correct answer, $2 + 8 = 10$, which would have been otherwise missed by temperature sampling.

local distribution over possible span continuations,

$$\mathbf{z} \sim p_\theta(\cdot \mid x_{1:t}, \#), \tag{2}$$

where $\mathbf{z}$ denotes a complete candidate span generated till the closing delimiter is encountered. SAGE operates on this span-level distribution rather than directly constraining the next-token distribution.

We will write $\mathrm{parse}(\mathbf{z})$ to denote the output of a parser that maps a mathematical span $\mathbf{z}$ to a set of formal claims. For a candidate span $\mathbf{z}_i$, we write

$$\Psi_i = \mathrm{parse}(\mathbf{z}_i), \tag{3}$$

where $\Psi_i$ corresponds to the set of claims: equalities, inequalities, implications, or other formalized mathematical relations, extracted from the span $\mathbf{z}_i$.

## 2.2 Satisfiability and SMT Solvers

We adopt the standard model-theoretic definition of *satisfiability* for first-order logic (Enderton, 2001). Let $\mathcal{L}$ be a logical language and let $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ be a finite set of formulas, then

**Definition 1** (Satisfiability)**.** *$\Phi$ is* satisfiable *if there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \varphi$ for all $\varphi \in \Phi$. Otherwise, we say that $\Phi$ is* unsatisfiable*.*

**Satisfiability Modulo Theories (SMT)** solvers check satisfiability of first-order formulas interpreted over background theories such as linear integer arithmetic, real arithmetic, bit-vectors, arrays, and strings. In this work, we primarily use real arithmetic as the background theory T. We write

$$\mathrm{SAT}_{\mathrm{T}}(\Phi)$$

for the solver status obtained by checking whether $\Phi$ is satisfiable modulo T, with possible outcomes such as SAT, UNSAT, or UNKNOWN. We use the Z3 (de Moura and Bjørner, 2008) SMT backend.

**Example 1.** *Let*

$$\Phi = \{x + y = 5,\ x > 0,\ y > 0\} \tag{4}$$

*over integer arithmetic. Then $\Phi$ is satisfiable: for instance, we can see that the assignment $x \mapsto 2,\ y \mapsto 3$ satisfies every formula in $\Phi$. In contrast,*

$$\Phi' = \{x + y = 5,\ x + y = 7\} \tag{5}$$

*is unsatisfiable, since no assignment makes both equations simultaneously true. For a given LM-generated reasoning trace, the same kind of inconsistency can arise when mathematical claims introduced across steps are jointly unsatisfiable.*

**Definition 2** (Symbolic self-consistency)**.** *Let* $\mathbf{y}$ *be a reasoning trace, and let* $\Phi(\mathbf{y})$ *denote the set of formal claims extracted from the mathematical spans of* $\mathbf{y}$*. We say that* $\mathbf{y}$ *is* symbolically self-consistent *modulo background theory* $\mathtt{T}$ *if* $\Phi(\mathbf{y})$ *is satisfiable modulo* $\mathtt{T}$*. Otherwise, we say that the reasoning trace* $\mathbf{y}$ *is* symbolically self-inconsistent*.*

## 3 Satisfiability-Guided Decoding

### 3.1 Problem Formulation

Let $\mathbf{s}$ denote a problem statement, and let $\tilde{\mathbf{y}} \sim p_\theta(\cdot \mid \mathbf{s})$ be a reasoning trace sampled from an LM. The trace $\tilde{\mathbf{y}}$ consists of natural-language text interleaved with mathematical statements. We instruct the model to delimit mathematical statements using $\#\ldots\#$, allowing the decoder to identify mathematical span boundaries throughout the generation.

A generated reasoning trace may successfully dictate a promising high-level approach to a given problem, while being symbolically *self-inconsistent* in the sense given in Definition 2. In what follows, we posit an unobserved *self-consistent* trace $\mathbf{y}$ associated with the model's sampled trace $\tilde{\mathbf{y}}$. Intuitively, we wish to *project* $\tilde{\mathbf{y}}$ onto the solution space defined by the SMT solver, yielding a *self-consistent* reasoning trace that respects the spirit of the high-level approach (i.e., remains plausible w.r.t. the LM's distribution) while satisfying the symbolic constraints induced by the reasoning traces' own mathematical claims. In this light, satisfiability-guided decoding can be viewed as conditional inference over LM-generated reasoning traces: preserve the LM's distribution over plausible traces while conditioning on the joint satisfiability of the extracted mathematical claims.

### 3.2 Symbolic Consistency as Conditioning

Given a reasoning trace $\mathbf{y}$, let $\Phi(\mathbf{y})$ denote the set of formal claims extracted from its mathematical spans. A minimal symbolic *self-consistency* requirement is that these claims be jointly satisfiable:

$$\mathtt{consistent}(\mathbf{y}) = \mathbb{1}\{[\Phi(\mathbf{y})\ \mathtt{sat\ modulo\ T}]\}\,. \tag{6}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function. The ideal distribution over symbolically self-consistent reasoning traces can therefore be given by

$$p_\theta(\mathbf{y} \mid \mathbf{s}, \mathtt{consistency}) \tag{7}$$

$$= \frac{p_\theta(\mathbf{y} \mid \mathbf{s}) \cdot \mathtt{consistent}(\mathbf{y})}{\sum_{\mathbf{y}'} p_\theta(\mathbf{y}' \mid \mathbf{s}) \cdot \mathtt{consistent}(\mathbf{y}')}, \tag{8}$$

where $\mathtt{consistency} \coloneqq \{\mathbf{y} \mid \mathtt{consistent}(\mathbf{y})\}$ denotes the set of all consistent reasoning traces. This formalizes the objective of preserving the LM distribution as an imperfect reasoner over plausible reasoning traces while restricting attention to symbolically self-consistent reasoning traces.

Directly sampling from this full-trace conditional distribution is intractable. One might instead hope to enforce satisfiability using standard token-level constrained decoding. However, tokenwise constrained decoding requires a *next-token viability oracle*: given the current symbolic state $\Phi_t$ and a partial mathematical span, it must determine which tokens $v \in \mathbb{V}$ admit some future span completion $\boldsymbol{z}$ whose parsed claims remain satisfiable:

$$\exists \boldsymbol{z} \quad \mathrm{SAT}_{\mathtt{T}}\left(\Phi_t \cup \mathrm{parse}(v \circ \boldsymbol{z})\right) = \textsc{sat}. \tag{9}$$

This is feasible for many syntactic constraints, which can be compiled into automata, grammars, or incremental parsers. SMT-based consistency constraints do not expose this interface. An SMT solver can check whether a proposed set of formal constraints is satisfiable, but it does not efficiently characterize which of the $|\mathbb{V}|$ next tokens can eventually complete to a satisfiable mathematical claim.

### 3.3 Locally Constrained Span Resampling

Since sampling from the full-trace conditional distribution is intractable, SAGE instead factorizes the constraint along mathematical span boundaries, conditioning on satisfiability one span at a time. Suppose the LM has generated prefix $\mathbf{x}_{1:t}$, and let $\Phi_t$ be the symbolic state induced by the claims accepted so far. At a new math span, the LM induces a local distribution over complete candidate spans,

$$\mathbf{z} \sim p_\theta(\cdot \mid \mathbf{x}_{1:t}, \#).$$

A candidate span is parsed into claims $\Psi = \mathrm{parse}(\mathbf{z})$. Self-consistency entails adding these claims while still retaining satisfiability:

$$\mathtt{consistent}_t(\mathbf{z}) = \mathbb{1}\left[\mathrm{SAT}_{\mathtt{T}}\big(\Phi_t \cup \Psi\big) = \textsc{sat}\right],$$

yielding the *locally-constrained span distribution*

$$q_t(\mathbf{z}) \propto p_\theta(\mathbf{z} \mid \mathbf{x}_{1:t})\, \mathtt{consistent}_t(\mathbf{z}).$$

Using rejection sampling, one can sample spans

$$\mathbf{z}_1, \ldots, \mathbf{z}_K \sim p_\theta(\cdot \mid \mathbf{x}_{1:t}),$$

parse each span, reject inconsistent candidates, and select uniformly among the satisfiable candidates.

This yields *self-consistent* spans that are faithful to the target conditional LM distribution.

Hard rejection is brittle with small $K$: candidates may fail to parse, the SMT solver may return UNKNOWN, or a span may contain only partially checkable claims. We thus relax the binary consistency event into a soft satisfiability verifier. Figure 2 summarizes this span-resampling procedure.

During decoding, SAGE maintains a symbolic state $\Phi_t$ containing the claims extracted from previously selected mathematical spans. A candidate span $\boldsymbol{z}$, with extracted claims $\Psi = \mathrm{parse}(\boldsymbol{z})$, is locally compatible with the trace so far if

$$\mathrm{SAT}_{\mathrm{T}}(\Phi_t \cup \Psi) = \textsc{sat}.$$

### 3.4 A Soft Measure of Self-Consistency

The definition of *self-consistency* in Equation (6) treats each candidate span as either consistent or inconsistent with the current symbolic state. This is too brittle in practice because the verifier is incomplete: a consistent mathematical span may fall outside the supported parser fragment, mix formal and informal claims, trigger a solver timeout, or receive an UNKNOWN outcome. Thus, failure to verify a span need not prove it as inconsistent.

We therefore replace the binary indicator with a soft verifier $v_T(\Phi_t, \Psi_i) \geq 0$. The verifier assigns higher weight to spans that are satisfiable or locally valid, zero weight to spans that are explicitly contradictory, and lower positive weight to undecided or malformed spans. In this manner, we take into account a measure of the span's self-consistency in addition to its likelihood under the LM.

Given a set of spans $\mathbf{z}_1, \ldots, \mathbf{z}_K \sim p_\theta(\cdot \mid \mathbf{x}_{1:t})$, with $\Psi_i = \mathrm{parse}(\mathbf{z}_i)$, SAGE approximates the verifier-tilted distribution by resampling from the finite candidate set according to verifier weights:

$$\Pr(\mathbf{z}^\star = \mathbf{z}_i) = \frac{v_{\mathrm{T}}(\Phi_t, \Psi_i)}{\sum_{j=1}^{K} v_{\mathrm{T}}(\Phi_t, \Psi_j)}.$$

The LM probability enters through the proposal step: high-probability spans are more likely to appear among the samples. The verifier then contributes only the local compatibility weight. In the hard-verifier case, this reduces to uniform resampling among sampled candidates that preserve satisfiability. We also consider a deterministic variant that selects the most compatible candidate

$$\mathbf{z}^\star \in \arg\max_{\mathbf{z}_i} v_{\mathrm{T}}(\Phi_t, \Psi_i),$$

**Algorithm 1** Satisfiability-Guided Decoding

**Require:** LM $p_\theta$, prompt $\mathbf{s}$, parser parse, verifier $v_{\mathrm{T}}$, candidates $K$

$x \leftarrow \mathbf{s}$, $\Phi \leftarrow \emptyset$
**while** not stopped **do**
  Generate tokens until <EOS> or delimiter
  **if** <EOS> reached **then return** $x$
  **end if**
  Sample spans $\mathbf{z}_1, \ldots, \mathbf{z}_K \sim p_\theta(\cdot \mid x)$
  **for** $i = 1, \ldots, K$ **do**
    $\Psi_i \leftarrow \mathrm{parse}(\mathbf{z}_i)$
    $w_i \leftarrow v_{\mathrm{T}}(\Phi, \Psi_i)$
  **end for**
  **if** $\sum_i w_i > 0$ **then**
    Sample $\mathbf{z}^\star$ using weights $w_i$
  **else**
    Select $\mathbf{z}^\star$ using diagnostic fallback
  **end if**
  $x \leftarrow x \circ \# \circ \mathbf{z}^\star \circ \#$
  $\Phi \leftarrow \mathrm{Update}(\Phi, \mathrm{parse}(\mathbf{z}^\star))$
**end while**
**return** $x$

with ties broken by diagnostic priority or local model score. This deterministic rule is not an unbiased sampler from $q_t$; rather, it is a mode-seeking approximation that favors the most solver-compatible continuation among the LM's local proposals. This distinction separates two uses of the same verifier: faithful constrained sampling from the locally conditioned span distribution, and accuracy-oriented constrained decoding.

### 3.5 The SAGE Algorithm

Algorithm 1 summarizes the common decoding framework underlying the two SAGE variants. Generation proceeds normally outside mathematical spans. When the model opens a span, SAGE samples $K$ complete candidate spans, parses each into claims, scores each candidate with the verifier $v_{\mathrm{T}}$, resuming generation from a compatible candidate.

### 3.6 Inference-Time Overhead

The overhead of SAGE is localized to mathematical spans. Otherwise, decoding is identical to standard LM generation. Let $N$ be the number of tokens in the selected reasoning trace, $L_{\mathrm{math}}$ the number of those tokens that fall inside mathematical spans, and $K$ the number of candidate spans generated at each fork. If $C_{\mathrm{LM}}$ is the cost of one LM decoding step, then standard decoding costs $O(NC_{\mathrm{LM}})$,

while SAGE incurs a cost of

$$O((N - L_{\text{math}}) \cdot C_{\text{LM}} + K \cdot L_{\text{math}} \cdot C_{\text{LM}}) .$$

Thus, the additional LM cost is

$$O((K - 1) \cdot L_{\text{math}} \cdot C_{\text{LM}}) .$$

Equivalently, let $\rho = L_{\text{math}}/N$ be the fraction of tokens inside mathematical spans, then we have that the idealized compute ratio is approximately

$$\frac{C_{\text{SAGE}}}{C_{\text{base}}} \approx 1 + (K - 1)\rho,$$

up to parser and solver overhead.

The parser and SMT solver add little GPU cost. Parsing is lightweight and roughly linear in span length, while Z3 checks are small CPU-side queries over the candidate claims and accumulated symbolic state. These calls do not consume GPU memory or GPU compute, so the dominant practical overhead is sampling the extra LM spans. Since typical reasoning traces contain relatively few delimited mathematical spans, $L_{\text{math}} \ll N$ in practice, and the overhead is substantially smaller than multiplying the full decoding cost by $K$.

## 4 Experiments

We empirically evaluate the effectiveness of the proposed approach across mathematical reasoning benchmarks and open-source instruction-tuned models, comparing standard temperature sampling to two variants of our method.

**Models.** We evaluate four open-source instruction-tuned LLMs across model families: LLaMA-3.2-3B-Instruct (Meta, 2024), LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Qwen3-4B-Instruct (Yang et al., 2025), and DeepSeek-Math-7B-Instruct (Shao et al., 2024).

**Methods.** We compare against two baselines: standard temperature sampling[1] and compute-matched majority voting. Because solver cost is negligible relative to generation, we measure SAGE's overhead as the total number of LM-generated tokens, including the final response and discarded candidate spans sampled at mathematical forks. For a conservative compute-matched majority-voting baseline, we round this overhead up to the next odd integer and use that many complete solutions. We discuss the incompatibility of using SATLM as a baseline in Appendix F.

[1]HuggingFace implementation

We evaluate two variants of SAGE: SAGE-`Rank`, which ranks candidate continuations using both solver feedback and model likelihood, and SAGE-`Sample`, which samples from a solver-weighted distribution over candidates. Unless otherwise specified, SAGE denotes SAGE-`Rank`, the default variant used in the main comparison.

**SAGE-Rank**. At each step, prose tokens (non-math text) are generated via top-$k$ sampling. When the model emits a math delimiter, generation forks: one candidate span is generated greedily and $N - 1$ additional candidates are generated via top-$k$ sampling. All $N$ candidates are evaluated by a Z3 solver, which parses all equations from the prefix and candidate and checks their logical consistency. If the greedy candidate passes the solver check (i.e., its status is satisfiable, valid, or produces no constrainable equations), it is retained unchanged. Otherwise, the remaining candidates are ranked jointly by solver status priority (Table 7) and average token log-probability, and the top-ranked candidate is selected.

**SAGE-Sample**. Prose tokens are generated identically to SAGE-Rank via top-$k$ sampling. At each math delimiter, all $N$ candidates are generated by sampling — there is no greedy anchor. Each candidate is evaluated by the same Z3 solver. Rather than rank-based selection, a candidate is drawn randomly with probability proportional to its solver status priority score (Table 7).

For temperature sampling and both variants of SAGE, we generate five samples per problem using matched random seeds, with temperature $T = 1$ and top-$k = 20$. We construct the majority-voting baselines from subsets of the temperature-sampled solutions. For both SAGE variants, we additionally generate five candidate continuations at each mathematical expression span. We use the same system prompt in all settings, which specifies that mathematical expressions should be enclosed in # signs; these expression delimiters are required by SAGE for equation-span sampling and extraction.

**Benchmarks.** We evaluate on three mathematical reasoning benchmarks. **MATH-500** ($n = 500$) is a curated 500-problem subset of MATH (Hendrycks et al., 2021), spanning seven categories (Algebra, Prealgebra, Number Theory, Geometry, Counting & Probability, Intermediate Algebra, Precalculus) and five difficulty levels. **GaoKao** ($n = 385$) is based on China's National College Entrance Examination and contains challenging high-school-

| Dataset | Model | Accuracy | | Pass@2 | | Pass@3 | |
|---|---|---|---|---|---|---|---|
| | | Sampling | SAGE | Sampling | SAGE | Sampling | SAGE |
| MATH-500 | LLaMA-8B | 33.4 | **39.7** | 46.6 | **51.2** | 53.6 | **57.0** |
| | LLaMA-3B | 37.5 | **39.9** | 48.0 | **50.5** | 53.6 | **56.3** |
| | DeepSeek-7B | 35.6 | **38.6** | 46.7 | **50.0** | 53.0 | **55.8** |
| | Qwen3-4B | 52.6 | **53.2** | 57.2 | **57.7** | 59.6 | **60.0** |
| Gaokao | LLaMA-8B | 32.4 | **35.1** | 41.8 | **44.2** | 46.9 | **48.9** |
| | LLaMA-3B | 31.6 | **32.8** | 41.4 | **42.4** | 46.8 | **47.6** |
| | DeepSeek-7B | 33.2 | **35.4** | 43.9 | **45.2** | 49.5 | **49.9** |
| | Qwen3-4B | 47.4 | **48.0** | 50.4 | **50.6** | 51.6 | **51.6** |
| Minerva | LLaMA-8B | 15.0 | **15.8** | 23.0 | **25.3** | 28.6 | **31.6** |
| | LLaMA-3B | 13.2 | **14.8** | 21.1 | **23.5** | 26.9 | **29.2** |
| | DeepSeek-7B | 19.4 | **20.3** | 29.2 | **30.1** | 35.2 | **36.5** |
| | Qwen3-4B | **24.6** | 24.4 | 29.0 | **29.6** | 31.7 | **32.8** |

Table 1: **Performance of Sampling and SAGE across three benchmarks.** Accuracy is the mean over seeds. Pass@$k$ ($k \in \{2, 3\}$) uses the unbiased estimator over seeds; **bold** = best between the two methods within each metric. LLaMA-8B and LLaMA-3B correspond to the 3.1 and 3.2 model families, respectively.

level mathematics problems requiring algebraic, geometric, symbolic, and multi-step quantitative reasoning (Zhang et al., 2023). **Minerva** ($n = 272$) is a collection of university-level quantitative reasoning problems introduced by Lewkowycz et al. (2022), emphasizing advanced STEM reasoning across mathematics, physics, engineering, and related technical domains.

**Metrics.** We report two evaluation metrics. First, **accuracy** measures whether the model's final answer matches the ground-truth answer. Because each method generates five samples per problem, we report accuracy averaged across samples. Second, **pass@$k$** measures the probability that at least one of $k$ sampled solutions is correct, estimated using the unbiased estimator (Chen et al., 2021). For a problem with $n$ generated samples, of which $c$ are correct,

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}},$$

with the value defined as 1 when $n - c < k$.

### 4.1 Results

Table 1 compares SAGE against standard temperature sampling across three mathematical reasoning benchmarks and four instruction-tuned language models. Overall, SAGE consistently improves over standard sampling across both average accuracy and Pass@$k$, indicating that solver-guided continuation selection improves not only the expected correctness of individual samples but also the probability of obtaining at least one correct solution among multiple attempts.

The largest improvements occur on MATH-500, where SAGE improves every model and metric, with average gains of 3.1% in accuracy, 2.7% in Pass@2, and 2.3% in Pass@3. The strongest gains are observed for LLaMA-8B, which improves by 6.3% accuracy points, 4.6% Pass@2 points, and 3.3% Pass@3 points. A finer-grained breakdown shows that the improvements persist across subject areas (Table 8). By difficulty level, the largest gains are concentrated at intermediate difficulty levels (L2–L3) across all models: LLaMA-8B improves by 11.6% at L2 and 8.8% at L3, LLaMA-3B by 5.1% and 3.3%, DeepSeek-7B by 3.8% and 7.7%, and Qwen's highest gains at L2 of 1.8% (Figure 8).

On the harder GaoKao and Minerva datasets, gains are consistent but more modest. The highest gains on GaoKao were for LLaMA-8B, which improves by 2.7% on accuracy, and 2.5% on Pass@2. On Minerva, the highest accuracy gain was 1.5% for LLaMA-3B, 2.4% for pass@2 for LLaMa-3B, and 3% for pass@3 for LLaMa-8B. The only accuracy regression is a small $-0.1$ point change for Qwen3-4B on Minerva, while its Pass@2 and Pass@3 still improve by 0.5% and 1.2%.

**Compute-matched majority voting.** Across models, datasets, and seeds, SAGE incurs an average token-generation overhead of approximately $2\times$. Because a strict majority vote requires an odd number of samples, we compare against majority voting over three complete solutions, giving the baseline a generous $3\times$ compute budget. Averaged over all three-sample subsets of the temperature-sampled solutions, SAGE outperforms MV@3 by 3.1 percentage points, with improvements observed

| Dataset | Model | MV@3 | SAGE | Δ |
|---|---|---|---|---|
| MATH | LLaMA-8B | 32.5±1.7 | **39.7±1.8** | +7.2 |
| | LLaMA-3B | 37.0±1.4 | **39.9±1.8** | +2.8 |
| | DeepSeek-7B | 34.3±1.3 | **38.6±1.2** | +4.2 |
| | Qwen3-4B | 52.2±0.6 | **53.2±0.8** | +1.0 |
| Gaokao | LLaMA-8B | 31.5±0.9 | **35.1±2.3** | +3.6 |
| | LLaMA-3B | 30.6±1.3 | **32.8±0.7** | +2.2 |
| | DeepSeek-7B | 32.6±1.0 | **35.4±2.5** | +2.8 |
| | Qwen3-4B | 47.9±0.6 | **48.0±0.7** | +0.1 |
| Minerva | LLaMA-8B | 11.7±1.0 | **15.8±1.9** | +4.2 |
| | LLaMA-3B | 9.4±1.1 | **14.8±2.3** | +5.4 |
| | DeepSeek-7B | 17.2±1.1 | **20.3±1.4** | +3.1 |
| | Qwen3-4B | 23.8±1.0 | **24.4±1.4** | +0.6 |

Table 2: **Compute-matched baseline.** Comparison of majority voting and SAGE, with results reported as accuracy ± standard deviation. Majority voting is given a generous 3× compute budget, compared with the approximately 2× generation overhead of SAGE. Δ denotes differential. **boldface** denotes higher accuracy.

in every dataset–model setting (Table 2). In 7 of the 12 dataset–model settings, the mean ± one-standard-deviation intervals do not overlap, suggesting that the gains are not attributable solely to sampling variability.

### 4.2 Effect of Candidate Selection Strategy

Table 6 compares two variants of SAGE against standard temperature sampling, reporting deltas relative to the sampling baseline. SAGE-Rank, our default, ranks candidates using solver feedback and model likelihood, whereas SAGE-Sample samples from a solver-weighted distribution over candidates.

Overall, SAGE-Rank performs substantially better: averaged across all model–dataset settings, it improves accuracy by 1.8%, Pass@2 by 1.8%, and Pass@3 by 1.7%, compared to 0.5%, 0.3%, and 0.2% for SAGE-Sample. SAGE-Sample occasionally performs better, most notably for DeepSeek-7B on MATH-500 and Minerva under Pass@$k$, but it also produces many more regressions, especially on Gaokao and Minerva, while SAGE-Rank is positive in nearly all cases.

### 4.3 Diagnostics

To understand the mechanisms behind SAGE's gains, we examine some diagnostic questions.

**What is the distribution of solver statuses?** Figure 3 shows the distribution of the solver status of the selected candidate at the final math-span fork, collapsing non-SAT/UNSAT outcomes such as unknown, parse errors, unsupported expressions, and missing equations into Unverified. Correct solutions consistently contain a higher fraction of SAT outcomes than incorrect solutions, while incorrect solutions tend to have higher UNSAT rates, indicating that solver verification provides a meaningful correctness signal.

| Position | SAT | UNSAT | Unverified |
|---|---|---|---|
| Early | 6.9 | 29.0 | 64.1 |
| Mid | 2.5 | 39.2 | 58.3 |
| Late | 1.2 | 43.6 | 55.3 |

Table 3: Verification outcomes of overridden forks by relative position in the solution trace. Earlier corrections are more likely to yield a SAT candidate, whereas later corrections are more frequently UNSAT.

The signal is weaker on Minerva. For Qwen3-4B, the pattern even reverses, coinciding with the observed regression in this setting. This suggests that the current parser and/or solver may be less compatible with Qwen3-4B generated reasoning chains over Minerva's scientific content.

The largest category overall is Unverified often accounting for 36% - 63% of outcomes, showing that many generated reasoning steps fall outside the solver's coverage. Since our results show a meaningful signal between successfully parsed solver statuses and correctness, future work can build on SAGE by enriching our simple rule-based parser and integrating solvers with broader coverage.

**What is the average number of math spans per solution?** Across models and datasets, the number of math spans per solution averages 12.4.

**How often does SAGE change or reject the greedy candidates?** 62.1% across models and datasets, so it is a common occurrence. This indicates that SAGE frequently departs from the greedy span, so the gains cannot be explained solely by always retaining a greedy math-span candidate.

**What are the positions of corrections?** We define the relative position of fork $i$ in a solution trace containing $F$ forks as $i/F$. We divide this measure into terciles: *early* (first third), *mid* (middle third), and *late* (final third). Pooled across settings, overridden forks occur 25.0% early, 36.1% mid, and 38.9% late, indicating that corrections are modestly concentrated toward the end of the trace. However, earlier corrections are more likely to result in a SAT candidate than later ones (Table 3).

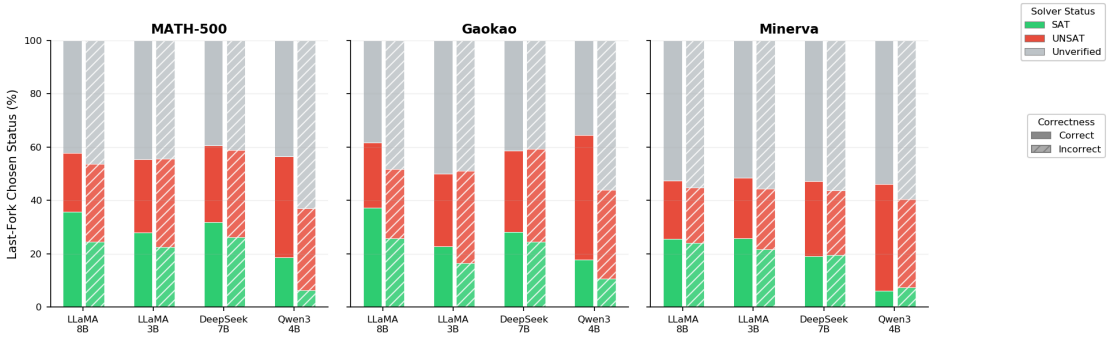



Figure 3: **Solver Status.** Distribution of solver statuses at the final math-span fork for correct and incorrect solutions. Non-SAT/UNSAT outcomes are grouped as *Unverified*. Correct solutions generally have higher SAT rates, whereas incorrect solutions have higher UNSAT rates, suggesting that solver status is informative of correctness.

## 5 Related Work

**Augmenting LMs with solvers.** Zhou et al. (2024) autoformalize complete LM-generated math solutions and verify them with a theorem prover, retaining verified solutions for majority voting. SATLM (Ye et al., 2023) instead translates the input problem into a declarative satisfiability specification that is solved by Z3, while He-Yueya et al. (2023) interleave natural-language reasoning with variables and equations handled by symbolic equation solvers. Related approaches use theorem provers or symbolic inference engines for logical reasoning (Olausson et al., 2023; Pan et al., 2023). These methods either verify completed solutions or use the LM to produce a solver-readable representation. In contrast, SAGE uses the solver as an *online* decoding signal, checking intermediate mathematical claims against the accumulated symbolic state.

**Controllable generation.** Our work also relates to methods that steer LM generation toward syntactic, logical, or semantic constraints by modifying the decoding distribution (Ahmed et al., 2023, 2025b; Zhang et al., 2024; Geh et al., 2024; Willard and Louf, 2023; Koo et al., 2024; Ahmed et al., 2025a; Yidou-Weng et al., 2025; Zhao et al., 2024; Yang and Klein, 2021; Dathathri et al., 2020; Liu et al., 2021, *inter alia*). SAGE is particularly close in spirit to locally constrained resampling (Ahmed et al., 2025b): it samples local continuations from the LM and uses an external constraint to restrict their selection. The key distinction is the constraint interface: SAGE conditions complete mathematical spans on compatibility with the symbolic state accumulated during the reasoning trace.

**Constrained decoding.** Constrained decoding methods enforce syntactic, lexical, or semantic properties during generation (Scholak et al., 2021; Lu et al., 2021, 2022; Poesia* et al., 2022; Agrawal et al., 2023). Recent work extends this paradigm to reasoning. Logically Constrained Decoding (Ma and Hu, 2025) restricts token-level continuations using formal domain rules, while CRANE (Banerjee et al., 2025) alternates between unconstrained reasoning and constrained generation of formal outputs. SAGE instead preserves free-form generation and applies SMT checks at mathematical-span boundaries, constraining the compatibility of successive mathematical claims with the symbolic state induced by earlier claims rather than the validity of individual tokens w.r.t. a formal language.

## 6 Conclusion

We introduced SAGE, a satisfiability-guided decoding method for improving the consistency of mathematical reasoning chains. Rather than verifying complete solutions post hoc or constraining every token, SAGE applies symbolic checks at equation-span boundaries, favoring continuations consistent with accumulated constraints. Across three mathematical reasoning benchmarks, this approach improves sampling-based performance across multiple model families, showing that solver guidance can make free-form reasoning chains more globally consistent. In addition, SAGE outperforms majority voting in a compute-matched setting, indicating that the gains are not solely due to extra generation budget. Final-fork solver status (SAT vs. UNSAT) also correlates with correctness, suggesting that solver outcomes provide a meaningful correctness signal. SAGE is limited by the coverage of the rule-based parser and solver-supported mathematical fragments. Extending to context-sensitive parsing and broader solver theories is promising.

## Limitations

**Z3 Solver** Our method can only enforce constraints that the backend solver can express and decide reliably. Although Z3 supports many useful theories, including arithmetic, bit-vectors, arrays, datatypes, strings, and uninterpreted functions, it is not a complete decision procedure for arbitrary mathematics. Some reasoning patterns that appear in natural-language solutions—for example, higher-order arguments, induction, geometry-specific facts, combinatorial reasoning, or difficult nonlinear/integer constraints—may fall outside the supported or practically decidable fragments. In such cases, the solver may fail to prove inconsistency, return `unknown`, or require encodings beyond the scope of our method. Consequently, our constraints improve local mathematical consistency but do not amount to full formal verification of the solution.

**Rule-Based Parser.** The effectiveness of our constrained decoding procedure also depends on correctly extracting and parsing equation spans from generated text. We use a deterministic rule-based parser, which avoids introducing an additional neural formalization step but is limited to expressions that can be recognized from surface form. This parser can miss mathematical statements that require semantic understanding of context, implicit quantities, units, or references to earlier text. It may also fail on unusual notation, underspecified variable bindings, or reasoning steps expressed in prose rather than explicit equations. Consequently, our method only constrains the parts of the generation that are successfully extracted and parsed, leaving other forms of mathematical reasoning unchecked.

## Acknowledgments

The authors thank the UCI NLP group for helpful discussions. This work is supported by DARPA ANSR program FA8750-23-2-0004, and an NSF CAREER award number IIS-2046873. The conclusions are of the authors and do not reflect the official policy or position of DARPA or the U.S. Government.

## References

Lakshya A Agrawal, Aditya Kanade, Navin Goyal, Shuvendu K. Lahiri, and Sriram K. Rajamani. 2023. Monitor-guided decoding of code lms with static analysis of repository context. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Kareem Ahmed, Catarina G. Belém, Padhraic Smyth, and Sameer Singh. 2025a. Semantic probabilistic control of language models. *CoRR*, abs/2505.01954.

Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. 2023. A pseudo-semantic loss for autoregressive models with logical constraints. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. 2025b. Controllable generation via locally constrained resampling. In *The Thirteenth International Conference on Learning Representations*.

Anthropic. 2025. Claude code. `https://github.com/anthropics/claude-code`. Accessed 2026-05-24.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. 2025. CRANE: Reasoning with constrained LLM generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 2836–2857. PMLR.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 34 others. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 2008. Z3: An efficient smt solver. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Kenneth Heafield, Onur Çelebi, Whitney Meers, Francisco Guzmán, and 1 others. 2024. The Llama 3 herd of models. *arXiv*, abs/2407.21783.

Herbert B. Enderton. 2001. *A Mathematical Introduction to Logic*, 2nd edition. Academic Press.

Renato Lui Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang, and Guy Van den Broeck. 2024. Where is the signal in tokenization space? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Joy He-Yueya, Gabriel Poesia, Rose Wang, and Noah Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Terry Koo, Frederick Liu, and Luheng He. 2024. Automata-based constraints for language model decoding. *Preprint*, arXiv:2407.08103.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Franklin Ma and Alan J. Hu. 2025. Logically constrained decoding. In *Proceedings of the Workshop on Mathematical Natural Language Processing (MathNLP)*.

Meta. 2024. Llama 3.2 model card. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/. Accessed 2026-05-22.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

OpenAI. 2022. Chatgpt. https://chat.openai.com/. Accessed 2026-05-24.

OpenAI. 2025a. GPT-5 system card.

OpenAI. 2025b. Introducing codex. https://openai.com/index/introducing-codex/. Accessed 2026-05-24.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Gabriel Poesia*, Alex Polozov*, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junmei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satlm: satisfiability-aided language models using declarative prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Gwen Yidou-Weng, Benjie Wang, and Guy Van den Broeck. 2025. Trace back from the future: A probabilistic reasoning approach to controllable language generation. In *Proceedings of the 42nd International Conference on Machine Learning*.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Nanyun Peng, and Guy Van den Broeck. 2024. Adaptable logical control for large language models. In *Advances in Neural Information Processing Systems*.

Xiaotian Zhang, Chun yan Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *ArXiv*, abs/2305.12474.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. 2024. Probabilistic inference in language models via twisted sequential monte carlo. In *Proceedings of the 41st International Conference on Machine Learning*.

Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. 2024. Don't trust: Verify–grounding llm quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*.

## A Computational Resources

All experiments were carried out on a cluster of NVIDIA RTX A6000 (48 GB VRAM) GPUs.

## B AI Assistant Usage

The authors used AI assistance in a limited capacity for implementation support and paper writing. Claude Code (Anthropic, 2025) and Codex (OpenAI, 2025b) were used primarily for debugging, refactoring, and localized implementation suggestions. ChatGPT (OpenAI, 2022) was used for writing assistance, limited to editing. All research ideas, experiments, analyses, claims, and final text were reviewed and approved by the authors, who take full responsibility for the content of the paper.

## C Inconsistencies in Reasoning Chains

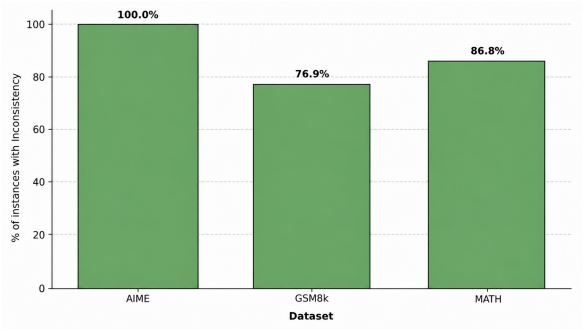



Figure 4: **Solution-level inconsistency prevalence.** Among solutions with incorrect final answers, more than 75% contain at least one inconsistency across the three benchmarks.

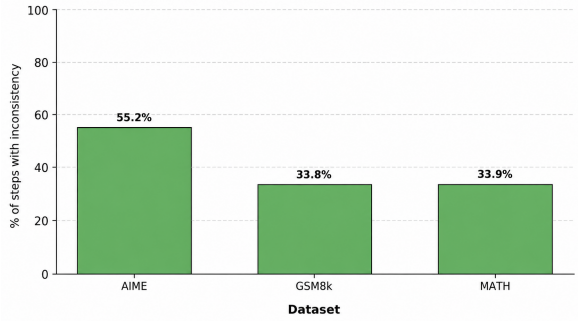



Figure 5: **Step-level inconsistency prevalence.** Across datasets more than 30% of steps in incorrect solutions contain some type of inconsistency.

To motivate SAGE, we briefly examine how frequently inconsistencies occur in the reasoning chains produced by LLMs for mathematical problems. We use the open-source Llama-3.1-8B-Instruct model to generate solutions to problems from AIME 2023 ($n = 15$), MATH ($n = 500$), and

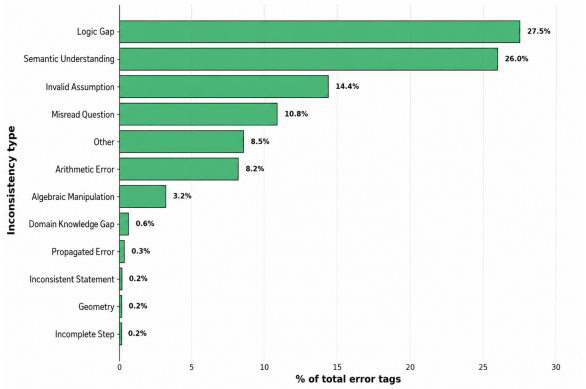



Figure 6: **Distribution of inconsistency types.** Logic gaps and semantic misunderstandings are the most common inconsistency types, accounting for 27.5% and 26.0% of all error tags, respectively.

| | |
|---|---|
| **Question** | Patty's Plumbing charges $40 to visit a house to make a repair, plus $35 per hour, or part thereof, for labor, plus parts. One job took 2.25 hours and used $60 in parts. How much did Patty charge? |
| **LLM solution** | **Step 1: Calculate the labor cost.** The labor cost is $35 per hour, and the job took 2.25 hours. Thus, the labor cost is $$\$35 \times 2.25 = \$78.75.$$ **Step 2: Add the labor cost, parts, and visit charge.** The total charge is the sum of the labor cost, parts, and visit charge: $$78.75 + 60 + 40 = 178.75.$$ Therefore, the final answer is $\boxed{\$178}$. ✗. |

Table 4: **Semantic misunderstanding.** An example of a case where the LLM final answer is incorrect and there is also an inconsistency within the solution. The model fails to account for the phrase "per hour, or part thereof," and therefore bills 2.25 hours rather than rounding the labor time up to three hours.

a random sample of 100 problems from GSM8K ($n = 100$). We then use GPT-5 (OpenAI, 2025a) to annotate the reasoning chains of solutions with incorrect final answers, identifying whether each solution contains an inconsistency and the reasoning steps in which inconsistencies occur.

We find that inconsistencies are common across all evaluated datasets. On AIME every solution with an incorrect final answer contains at least one inconsistency, while the corresponding proportion exceeds 75% on the remaining datasets (Figure 4). At a finer level of granularity, more than 30% of reasoning steps across datasets contain some form of inconsistency (Figure 5).

The inconsistencies also take several forms. Across datasets, 27% of reasoning steps contain gaps in logic (Figure 6). Another common category, accounting for 26% of inconsistent cases, involves a semantic misunderstanding of either the problem statement or an intermediate reasoning step (Figure 6). Other observed inconsistencies include arithmetic errors and incorrect algebraic manipulations.

Table 4 presents an example of a semantic inconsistency. The problem asks for the total amount charged by Patty's Plumbing, which includes a flat visit fee, an hourly labor fee, and the cost of parts. The key detail is that labor is billed "per hour or part thereof," meaning that any fractional hour must be rounded up to the next full hour. Because the job takes slightly more than two hours, the customer should be billed for three hours of labor. However, the generated solution directly uses 2.25 hours in its first step, leading to an incorrect final answer.

Inconsistencies can also occur in reasoning chains whose final answers are correct. Table 5 shows an example for the following problem: *What is the smallest positive perfect cube that can be written as the sum of three consecutive integers?*

A correct solution first represents the three consecutive integers algebraically using an integer variable, say $m$. Their sum is then equated to a perfect cube, say $n^3$. Because the resulting cube must be divisible by three, $n$ must itself be divisible by three. The smallest valid value is therefore $n = 3$, yielding $n^3 = 27$. The model-generated solution arrives at the same final answer, but nevertheless contains several inconsistencies. It initially uses the same variable, $n$, to represent both the consecutive integers and the cube root. These quantities are distinct: in the correct derivation, the variable representing the consecutive integers takes the value $m = 9$, whereas the cube root is $n = 3$. The remainder of the reasoning contains additional errors, including incorrectly concluding that $n^2 = 3$ implies $n = 3$ or $n = -3$. Despite these inconsistencies, the model ultimately produces the correct final answer, 27. Standard answer-based evaluation would mark this solution as correct because its final answer is correct. This can overestimate model performance and create serious risks in high-stakes settings, including education, scientific discovery, and military applications.

**Question:** What is the smallest positive perfect cube that can be written as the sum of three consecutive integers?

**Correct solution**

**Step 1: Express the sum of three consecutive integers algebraically.**
Let the three consecutive integers be $m-1$, $m$, and $m+1$. Their sum is

$$(m-1)+m+(m+1)=3m.$$

**Step 2: Relate this to a perfect cube.**
We are looking for the smallest positive perfect cube, say $n^3$, that can be written in this form. Thus, we require

$$3m=n^3.$$

This shows that $n^3$ must be divisible by 3.
**Step 3: Determine when a cube is divisible by 3.**
A cube $n^3$ is divisible by 3 if and only if $n$ is divisible by 3. Thus, the smallest positive integer $n$ that makes $n^3$ divisible by 3 is $n=3$.
**Step 4: Compute the cube.**
Therefore,

$$n^3=3^3=27.$$

**LLM-generated solution**

**Step 1: Express the sum of three consecutive integers algebraically.**
Let the smallest positive perfect cube be $n^3$. We can express three consecutive integers as $n-1$, $n$, and $n+1$. The sum of these three consecutive integers can be written as

$$(n-1)+n+(n+1)=3n.$$

**Step 2: Solve for $n$.**
We are given that the sum is a perfect cube. Thus,

$$3n=n^3.$$

Dividing both sides by $n$ gives

$$3=n^2.$$

**Step 3: Solve for $n$.**
Since $3=n^2$, we can find the value of $n$ by taking the square root of both sides. However, since we are looking for the smallest positive value, we only consider the positive square root.
**Step 4: Solve for $n$.**
We know that $\sqrt{3}$ is not a perfect square, so we need to find a value of $n$ that satisfies $3=n^2$. This means we must round up to the nearest whole number that gives us 3 when squared.

**Step 5: Solve for $n$.**
The smallest whole number $n$ that satisfies the equation is $n=3$ or $n=-3$. But the smallest positive perfect cube will be the smallest value of $n^3$ where $n$ is positive. Therefore, the final answer is

$$3^3=27.$$

Table 5: **Correct answer with inconsistent reasoning.** The LLM arrives at the correct final answer despite the reasoning trace containing many inconsistencies, highlighted in red.

## D System Prompt

Figure 7 shows the system prompt used in all experiments in Section 4. The prompt specifies that complex problems should be solved step-by-step, all mathematical expressions should be enclosed in # signs, and the final answer should be reported in a LaTeX box.

## E Variants

We consider two variants of SAGE: SAGE-Rank, which ranks candidate continuations using both solver feedback and model likelihood, and SAGE-Sample, which instead samples candidates according to a solver-weighted distribution. Solver status priority scores used for ranking and selecting candidates can be found in Table 7.

## F SATLM

SATLM (Ye et al., 2023) is a neuro-symbolic reasoning approach that uses few-shot prompting to make an LLM produce a declarative parse of the input problem as a Z3-compatible constraint program. Rather than solving the problem directly, the LLM declares writes constraints relating them; Z3 then performs the reasoning by finding assignments that satisfy the resulting constraint system

**Setup.** We evaluate SAT-LM using the authors' original codebase with minimal modification: their OpenAI API calls are replaced with a local HuggingFace inference endpoint. We test Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, and DeepSeek-Math-7B-Instruct on MATH-500, Gaokao, and Minerva at temperatures $\tau \in \{0.6, 1.0\}$. All other settings—few-shot prompt, prompting style, and

| Dataset | Model | Accuracy | | Pass@2 | | Pass@3 | |
|---|---|---|---|---|---|---|---|
| | | Rank | Sample | Rank | Sample | Rank | Sample |
| MATH-500 | LLaMA-8B | **+6.3** | +2.5 | **+4.6** | +1.0 | **+3.3** | +0.1 |
| | LLaMA-3B | **+2.4** | +1.3 | **+2.5** | +1.6 | **+2.7** | +1.6 |
| | DeepSeek-7B | **+3.0** | +2.7 | +3.3 | **+3.5** | +2.9 | **+3.8** |
| | Qwen3-4B | **+0.6** | -0.5 | **+0.5** | -0.8 | **+0.4** | -0.7 |
| Gaokao | LLaMA-8B | **+2.7** | -0.6 | **+2.5** | -0.9 | **+2.0** | -1.3 |
| | LLaMA-3B | **+1.2** | +0.2 | **+1.0** | -0.3 | **+0.8** | -1.2 |
| | DeepSeek-7B | **+2.2** | +1.3 | **+1.3** | +0.1 | **+0.4** | -0.2 |
| | Qwen3-4B | **+0.5** | -0.1 | **+0.2** | -0.2 | **+0.0** | -0.1 |
| Minerva | LLaMA-8B | **+0.8** | +0.1 | **+2.3** | -0.5 | **+3.0** | -1.0 |
| | LLaMA-3B | **+1.5** | -0.7 | **+2.4** | -0.4 | **+2.3** | -0.3 |
| | DeepSeek-7B | **+0.9** | +0.2 | +0.9 | **+1.0** | +1.3 | **+2.1** |
| | Qwen3-4B | **-0.1** | -0.4 | **+0.5** | -0.4 | **+1.2** | -0.3 |

Table 6: **Variant Performance.** SAGE-Rank, and SAGE-Sample performance across three benchmarks. Accuracy is the mean over seeds. Pass@$k$ ($k \in \{2, 3\}$) uses the unbiased estimator over $n$=5 seeds. All values are deltas relative to Sampling, computed before rounding. LLaMA-8B and LLaMA-3B correspond to the 3.1 and 3.2 model families, respectively.

| Solver status | Score | Description |
|---|---|---|
| `sat` | 6 | All equations in the prefix are mutually satisfiable |
| `valid` | 6 | The claim is logically valid |
| `valid_equivalence` | 6 | Two expressions are proven equivalent |
| `valid_implication` | 6 | One expression is proven to imply another |
| `valid_chain` | 6 | A chain of equalities or inequalities is valid |
| `no_equations` | 4 | No constrainable equations found in the span |
| `ignored_fragment` | 4 | Span is too incomplete to form a checkable claim |
| `unknown` | 2 | Solver could not determine satisfiability |
| `parse_error` | 1 | Equation could not be parsed into a solver-readable form |
| `error` | 1 | Runtime error encountered during solving |
| `unsat` | 0 | Equations are mutually contradictory |
| `invalid_equivalence` | 0 | Two expressions are proven not equivalent |
| `invalid_implication` | 0 | The implication is proven not to hold |
| `invalid_chain` | 0 | A step in the equation chain is invalid |

Table 7: Solver status priority scores used for candidate ranking and weighted selection.

evaluation—are unchanged from the original paper. The 8-shot prompt consists of the math word problems from the original paper.

**Results.** Accuracy is 0% in every condition across all three datasets, models, and temperatures. The models failed to produce executable Z3-compatible specifications. Common failures include missing variable declarations and malformed constraints, causing the generated programs to fail before Z3 could solve them. This failure may reflect differences from the original SATLM setup: Ye et al. (2023) conduct their main experiments with `code-davinci-002`, a code-oriented model. Applying SATLM to smaller open-source instruction-tuned models may therefore require additional prompt engineering, stronger format instructions, or manually crafted exemplars tailored to each benchmark.

## G Subject and Difficulty Analysis

We analyse SAGE's performance on MATH-500, a 500-problem benchmark spanning seven problem types (Algebra, Counting & Probability, Geometry, Intermediate Algebra, Number Theory, Prealgebra, and Precalculus) and five difficulty levels (L1–L5). For each of the four models we compare Sampling and SAGE averaged over four to five random seeds, breaking results down both by difficulty level and by problem type to understand where the solver-guided decoding provides the most benefit.

Both Sampling and SAGE exhibit a consistent monotonic decrease in accuracy from L1 to L5 across all four models. On Sampling, accuracy ranges from 66–89% at L1 down to 11–23% at L5

**System Prompt**

Solve the following math problem efficiently and clearly:

**For simple problems (2 steps or fewer):**
Provide a concise solution with minimal explanation.

**For complex problems (3 steps or more):**
Use this step-by-step format:

```
## Step 1: [Concise description]
[Brief explanation and calculations]

## Step 2: [Concise description]
[Brief explanation and calculations]
```

Mathematical statements should be written in equations and enclosed in # signs.

. . .

Regardless of the approach, always conclude with:

Therefore, the final answer is: $\boxed{answer}$. I hope it is correct.

Where *[answer]* is just the final number or expression that solves the problem.

Figure 7: System prompt used for all model inference.

depending on the model. SAGE preserves this ordering while shifting the curves upward, but the magnitude of the gain is not uniform across levels. **Across all models, the largest gains tend to occur at intermediate difficulty levels L2 and L3.** LLaMA-8B improves by 11.6 pp at L2 and 8.8 pp at L3, LLaMA-3B by 5.1 pp at L2 and 3.3 pp at L3, and DeepSeek-7B by 3.8 pp at L2 and 7.7 pp at L3. Qwen3-4B, the strongest model, follows the same pattern but benefits least overall, with gains of at most +1.8 pp and near-zero or negative deltas at L4 and L5.

Table 8 breaks down accuracy by problem type on MATH-500. **SAGE improves over Sampling for the large majority of subjects across models.** The largest gains include LLaMA-8B on Counting & Probability (+13.2 pp) and LLaMA-8B on Algebra (+9.7 pp). Three small regressions appear: DeepSeek-7B on Prealgebra (-2.0 pp), LLaMA-3B on Precalculus (-1.4 pp), and Qwen3-4B on Number Theory (-0.3 pp).

**Competition-Level Math** Consistent with this pattern, on AIME 2023 (Balunović et al., 2025), LLaMA-3.1-8B-Instruct achieves 0% accuracy under both standard sampling and SAGE. This suggests that SAGE is most useful when the base model can reach a plausible reasoning trajectory: at floor performance, there are few recoverable paths for local symbolic correction to improve. Evaluating SAGE on AIME and other competition-level benchmarks with larger, frontier-scale models is an important direction for future work, but was beyond our compute budget.

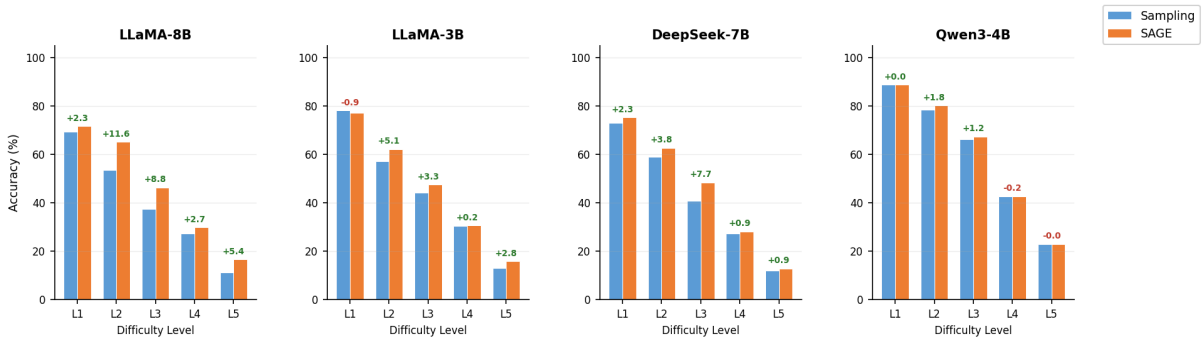



Figure 8: Accuracy (%) by difficulty level on MATH-500 for Sampling and SAGE, averaged across seeds.

| Subject | LLaMA-8B | | LLaMA-3B | | DeepSeek-7B | | Qwen3-4B | |
|---|---|---|---|---|---|---|---|---|
| | Sampling | SAGE | Sampling | SAGE | Sampling | SAGE | Sampling | SAGE |
| Algebra | 46.9 | **56.6** | 55.8 | **58.7** | 53.7 | **58.2** | 72.7 | **73.1** |
| C&P | 24.2 | **37.4** | 33.2 | **33.7** | 25.8 | **28.4** | 58.4 | **59.5** |
| Geometry | 28.8 | **30.2** | 26.8 | **29.3** | 29.3 | **33.2** | 42.9 | 42.9 |
| Int. Alg. | 14.8 | **19.4** | 17.7 | **18.6** | 14.6 | **19.2** | 21.0 | **22.9** |
| Num. Th. | 35.5 | **36.8** | 31.6 | **37.7** | 33.5 | **35.8** | **54.8** | 54.5 |
| Prealgebra | 50.2 | **56.8** | 55.1 | **58.8** | **55.9** | 53.9 | 75.6 | **75.9** |
| Precalculus | 18.2 | **23.6** | **22.9** | 21.5 | 15.3 | **19.6** | 30.2 | **30.2** |

Table 8: Accuracy (%) by problem type on MATH-500. **Bold** = best within each column pair.