# IS EXACT CONSTRAINED SAMPLING RIGHT FOR YOU?

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Constrained sampling lets a language model (LM) produce outputs that satisfy a grammar or another hard constraint, and how it is done matters: masking invalid tokens one step at a time distorts the LM's distribution over valid outputs, which harms applications that need that distribution, such as program fuzzing and molecule generation. Recent work samples from the correct constrained distribution either approximately or exactly, and the exact methods—adaptive rejection sampling (ARS) and the recent Constrained Adaptive Rejection Sampling (CARS)—are preferable when their cost is affordable. But nothing tells a practitioner in advance whether it is, or which exact method to use. We analyze the cost of these methods exactly. Over a run, ARS pays one rejected LM call for every possible first mistake of the LM, while CARS pays at most one per prefix, and CARS is optimal among all strategies that learn only from their own draws. The speedup of CARS over ARS is bounded below by two masses read off the trie CARS builds: the *light* invalid mass, too unlikely for ARS to have drawn, and the *unreachable* mass, at prefixes too rare for CARS to have reached. Our analysis is practical: the measured costs fall inside the predicted range on every task, tight when the run has seen most of the ways the LM violates the constraint.

## 1 INTRODUCTION

Many applications of Language Models (LMs) require outputs that are valid with respect to structural constraints: programs that parse, molecules in valid SMILES syntax, test inputs accepted by a grammar (Scholak et al., 2021; Geng et al., 2023; Melcer et al., 2024). When the samples must also follow the LM's own distribution conditioned on the constraint—as in program fuzzing or molecule discovery, where diversity matters as much as validity—constrained decoding (Park et al., 2025; Ugare et al., 2024; AI, 2025) is not enough: masking the invalid tokens at each step and renormalizing samples a different distribution, which can put much of its mass on outputs the LM itself considers unlikely (Park et al., 2024; Tam et al., 2024). Recent work therefore samples from the constrained distribution itself, approximately or exactly. Approximate methods—ASAp (Park et al., 2024), Markov chain Monte Carlo (Anaya Gonzalez et al., 2025), and sequential Monte Carlo (Lew et al., 2023; Loula et al., 2025; Lipkin et al., 2025)—reach the constrained distribution only in the limit of more steps or particles. Exact methods are rejection-based (Mansinghka et al., 2009; Parys et al., 2026): their samples follow the constrained distribution from the first sample on, and they differ in what they remember about a rejected draw.

Consider the grammar $E ::= d\$ \mid d{+}E$ over the digits $\{0, 1\}$, and an LM that has just produced `0++`. *Rejection sampling* (RS) discards the draw. *Adaptive Rejection Sampling* (ARS) records the shortest invalid prefix, `0++`, so that no future draw starts with it. *Constrained Adaptive Rejection Sampling* (CARS) queries the grammar at every prefix of the draw and records every token that was invalid there: every non-digit as a first token, everything but `+` and `$` after `0`, and everything but a digit after `0+`. All this information goes into a trie, and the mass of every recorded continuation is subtracted from future draws. These methods sample exactly from the constrained distribution, and for ARS and CARS the acceptance rate can only improve as the trie grows.

Exact sampling is clearly preferable when one can afford it, and its cost is almost entirely LM calls: in the CARS implementation, LM inference takes up most of the running time and the trie about $4\%$ (Parys et al., 2026). We therefore count cost in *LM calls*, each of which samples one complete sequence, per valid sample; and for an adaptive method the number of calls per valid sample decreases over the run. A practitioner who wants exact samples then faces two questions,



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

both theoretical: *(i)* **Speed:** how fast does the acceptance rate improve for ARS and CARS, that is, how many calls does each exact method need? *(ii)* **Scope:** which constraints, with which models, make the improvement large for each method, and when is exact sampling affordable at all?

What is known about either question is empirical. Measured across benchmarks, CARS's advantage over ARS ranges from marginal to more than an order of magnitude, and on some constraints no method of this kind makes progress at all. Existing theory stops at exactness, at monotone improvement of the acceptance rate, and at the qualitative ordering CARS $\geq$ ARS (Parys et al., 2026): CARS provably needs fewer samples, but by how much is left open, and its authors note that "the exact gain is hard to analyze analytically as it depends on both the grammar's branching structure and the LM's token distribution over invalid regions." *What is missing is a characterization of how many fewer samples CARS needs, and of when the gap is large, which a practitioner could use before paying for a run to decide whether exact sampling is worth it and which method to use.*

We give these characterizations for ARS and CARS. The key idea enabling these characterizations is the notion of *atom*: a pair $(v, a)$ of a grammatical prefix $v$—one that can still be completed validly—and a token $a$ that violates it, standing for the sequences whose first mistake is $a$ after $v$. Every prefix ARS or CARS rules out is the shortest invalid prefix of some sequence, so atoms partition the invalid mass, and the methods differ only in which draws remove an atom: ARS must draw the atom itself, CARS only some sequence through $v$, valid or not.

**Speed: how much faster is CARS than ARS?** Over an entire run, ARS pays one rejected call for every possible first mistake of the LM, however unlikely, while CARS pays at most one per prefix, and only when its first visit there goes wrong: in the running example of Example 2.1, 902 rejections against 2.9. Describing each method by the draws that remove an atom answers question *(i)* exactly. Imagining all strategies reading one shared stream of LM samples, each skipping the draws it has already excluded, makes the chance that an atom survives $n$ valid samples exact, and summing over atoms gives the expected number of rejected LM calls of CARS and ARS in closed form. The same shared stream shows that on every run CARS excludes at least as much as any rule that learns only from the prefixes of its own draws—ARS, or one that consults the oracle at only some positions. CARS is therefore optimal among all strategies that learn only from their own draws.

**Scope: when is CARS useful?** The same analysis answers question *(ii)*. After $k$ calls, the speedup of CARS over ARS is bounded below by two masses read off the trie CARS builds: the *light* mass, in atoms too unlikely for ARS to have drawn, and the *unreachable* mass, at prefixes too rare for CARS to have reached. The speedup is thus carried by invalid mass at prefixes the LM visits often, split over many unlikely tokens—in the example, the hundred light tokens at the root, not the heavy + after 0+, which ARS finds after a few valid samples. The two masses sort constraints into four kinds. At one end are format constraints, such as a mandatory keyword: every draw passes through the prefixes where the mistakes happen, so CARS clears them within its first few calls and then accepts almost every draw, while ARS still rejects at nearly the rate of rejection sampling—a speedup close to the ceiling $1/P(\mathcal{L})$, where $P(\mathcal{L})$ is the probability that an unconstrained sample from the LM satisfies the constraint. In between, tight per-position grammars put many light invalid tokens at every prefix, which CARS clears on its first visit and ARS rarely draws, while loose grammars and blacklists put a few heavy ones there, which ARS finds within a few valid samples, so CARS leads only slightly. At the other end are global constraints, violated only after a long and varied prefix: each mistake happens at a different rarely visited prefix, so neither CARS nor any other prefix-based method learns anything useful. This sorting is also the practical answer to question *(ii)*: on format constraints and tight grammars exact sampling with CARS is cheap, while global constraints make every prefix-based exact method expensive, and there an approximate method, or a prompt that moves the LM's mistakes earlier, is the better choice.

**Predicting the cost of a run.** Everything in the analysis except $P(\mathcal{L})$ can be read off the prefixes a run visits, so the theory can be checked against practice. On nine benchmarks of Parys et al. (2026), the prefixes visited by one CARS run predict how many LM calls independent runs of CARS and ARS need: the measured cost of CARS falls inside the predicted range on every task, and the range is tight. On four tasks, the predicted cost of producing 100 valid samples with CARS comes within 5% of the measured cost after only 3–10% of the calls those 100 samples take, so what a sampling job will cost can be known before paying for it, and with it whether to sample exactly at all.

**Algorithm 1** The common skeleton. RS, ARS and CARS differ only in INVALID.

$\mathcal{W} \leftarrow \emptyset$ ▷ *prefixes already known to be invalid*
**while** true **do**
  $w \sim Q^{\mathcal{W}}$ ▷ *draw from $P$ with $\mathcal{W}\Sigma^*\$$ removed and renormalized*
  **if** $w \in \mathcal{L}$ **then yield** $w$ ▷ *a sample from $P^{\mathcal{L}}$*
  $\mathcal{W} \leftarrow \mathcal{W} \cup \textsc{Invalid}(w, \mathcal{L})$ ▷ *learn from the draw*

## 2 RETHINKING ARS AND CARS THROUGH ATOMS

This section fixes the setting of Parys et al. (2026) and introduces the two notions every later result rests on: *atoms*, which split the invalid mass into the pieces the algorithms remove, and the *stream coupling*, which runs every strategy on the same draws.

Throughout, $\Sigma_\$$ is the token vocabulary including the end-of-sequence marker \$, complete sequences are the elements of $\Sigma^*\$$, and the LM assigns to a prefix $u \in \Sigma^*$ the probability $P(u)$ of generating it and to a token the conditional $P(ua \mid u)$. We write $u \prec w$ when $u$ is a proper prefix of $w$, and $u\Sigma^*\$$ for the complete sequences beginning with $u$ so that $P(u\Sigma^*\$) = P(u)$. A constraint is a set $\mathcal{L} \subseteq \Sigma^*\$$, and constrained sampling asks for draws from $P^{\mathcal{L}}(w) = P(w)/P(\mathcal{L})$. We call $P(\mathcal{L})$ the *valid mass* and $1 - P(\mathcal{L})$, the probability that the LM's sequence violates the constraint, the *invalid mass*. The *grammatical* prefixes $\mathrm{prefix}(\mathcal{L})$ are the prefixes of sequences in $\mathcal{L}$, those the LM can still complete to satisfy the constraint, and $P(\mathcal{L} \mid v) = P(v\Sigma^*\$ \cap \mathcal{L})/P(v)$ is the probability of completing validly from $v$. We assume, as Parys et al. (2026) do, a *prefix oracle*: given $u$ it reports, for every token $a$, whether $ua \in \mathrm{prefix}(\mathcal{L})$. Context-free grammars admit one.

**Three update rules.** All three algorithms are Algorithm 1. They keep a set $\mathcal{W}$ of prefixes known to be invalid, held as a trie, and draw from the proposal $Q^{\mathcal{W}}$: $P$ restricted to the sequences $\Sigma^*\$ \setminus \mathcal{W}\Sigma^*\$$ that extend no prefix in $\mathcal{W}$ (where $\mathcal{W}\Sigma^*\$ = \bigcup_{u\in\mathcal{W}} u\Sigma^*\$$), renormalized by their mass

$$p_\varepsilon(\mathcal{W}) = P\big(\Sigma^*\$ \setminus \mathcal{W}\Sigma^*\$\big).$$

Only the update INVALID differs. RS sets $\textsc{Invalid}(w, \mathcal{L}) = \emptyset$ and forgets the draw. ARS adds the shortest invalid prefix of $w$: for the draw `0++` of the introduction, the single prefix `0++`. CARS asks the oracle at every grammatical proper prefix $u$ of $w$ and adds every token that fails there, $\textsc{Invalid}(w, \mathcal{L}) = \{ua : u \prec w,\ u \in \mathrm{prefix}(\mathcal{L}),\ ua \notin \mathrm{prefix}(\mathcal{L})\}$; for `0++` that is the whole orange and blue fan at three prefixes of Figure 1a, not one edge. Because INVALID only ever adds prefixes outside $\mathrm{prefix}(\mathcal{L})$, the renormalization leaves the conditional distribution untouched: all three yield exactly $P^{\mathcal{L}}$. They differ only in how fast $p_\varepsilon$, the mass still in play, falls from 1 towards $P(\mathcal{L})$. A call is accepted with probability $P(\mathcal{L})/p_\varepsilon(\mathcal{W})$, so that fall is the whole question of speed.

**Atoms.** Every invalid sequence $w \notin \mathcal{L}$ has a unique shortest invalid prefix $va$: a grammatical prefix $v$ followed by a token $a$ with $va \notin \mathrm{prefix}(\mathcal{L})$. It marks the sequence's first mistake: up to $v$ it could still have been completed validly, and emitting $a$ ruled that out. The *atom* of $(v, a)$ is $A_{v,a} = va\Sigma^*\$$, the set of complete sequences whose first mistake is $a$ after $v$. Its mass is $P(va) = P(v)\,P(va \mid v)$; we write the product when the two factors play different roles. Atoms are disjoint and cover the invalid sequences, so $\sum_{(v,a)} P(va) = 1 - P(\mathcal{L})$.

**Bad tokens.** The *bad tokens* at $v$ are its invalid next tokens, and their atoms are all the atoms whose first mistake happens at $v$:

$$\mathrm{bad}_v = \{a : va \notin \mathrm{prefix}(\mathcal{L})\}, \qquad \mathrm{atoms}(v) = \bigcup_{a\in\mathrm{bad}_v} A_{v,a},$$

$$P(\mathrm{atoms}(v)) = \sum_{a\in\mathrm{bad}_v} P(va) = P(v)\,P(v\mathrm{bad}_v \mid v), \qquad P(v\mathrm{bad}_v \mid v) = \sum_{a\in\mathrm{bad}_v} P(va \mid v).$$

So $P(v\mathrm{bad}_v \mid v)$ is the probability that the LM's next token after $v$ is invalid, $|\mathrm{bad}_v|$ is the number of atoms at $v$, and $P(v)\,P(v\mathrm{bad}_v \mid v)$ is the *invalid mass at* $v$: the part of the invalid mass whose first mistake happens at $v$. Summed over all $v$ it is the whole invalid mass. A grammatical prefix with $P(v\mathrm{bad}_v \mid v) > 0$ is a *frontier node*: a place where the LM can make its first mistake.

**What a draw teaches.** ARS and CARS only ever add shortest invalid prefixes to $\mathcal{W}$, so $\mathcal{W}\Sigma^*\$$ is always a union of atoms, the state of either algorithm is the set of atoms it has removed, and

$$p_\varepsilon(\mathcal{W}) = P(\mathcal{L}) + \sum_{\text{atoms } (v,a) \text{ not removed}} P(va).$$

The two algorithms differ in which draws remove an atom. ARS removes $A_{v,a}$ only by drawing a sequence inside it. CARS removes the whole of $\mathrm{atoms}(v)$ as soon as it draws *any* sequence through $v$, valid or not, together with the bad tokens of every other prefix of that sequence. The draws that teach CARS about an atom $(v, a)$ thus have mass $P(v)$, against $P(va)$ for ARS: larger by the factor $1/P(va \mid v)$, and valid draws included. Section 3 turns this factor into a cost.

**One stream, every strategy.** To compare strategies on the same random draws rather than only on average, let every strategy read one i.i.d. stream $w_1, w_2, \ldots$ from $P$ and *skip* the samples in its excluded set $\mathcal{W}\Sigma^*\$$. Conditionally on the state, the first sample not skipped has law $Q^{\mathcal{W}}$, so the samples not skipped—the *LM calls*—form a run of the algorithm. Valid samples are never skipped, so the $n$-th valid sample sits at the same stream index $T_n$ for every strategy; this *stream coupling* is why Section 3 counts in valid samples. Write $\mathrm{path}(w)$ for the grammatical proper prefixes of $w$, its *path*: the prefixes at which it could have gone wrong, and so the prefixes whose bad tokens CARS learns when it draws $w$. The skipped samples are *inert*—skipping them costs nothing and reveals nothing—since a skipped sample only ever lies in a region already removed: after any number of stream samples CARS has removed exactly the atoms at the nodes on their paths, and ARS exactly the atoms containing one of them, whether or not the samples were skipped (Lemma A.1).

**Example 2.1** (Running example)**.** Figure 1a shows the example: the grammar $E ::= d\$ \mid d + E$ of the introduction, with $d \in \{\texttt{0}, \texttt{1}\}$, and an LM in the spirit of Figure 2 of Parys et al. (2026). The vocabulary has `0`, `1`, `+`, `$` and 100 other tokens, and the LM never continues past a second digit. The probabilities are chosen so that they read as counts out of 1,000 draws of the LM: 300 go wrong at the root, 500 begin with `0` and 200 with `1`; of the 500, 200 go wrong at `0`, 100 stop there validly and 200 continue with `+`; and so on down the figure. The draws that go wrong at a node are spread evenly over the tokens invalid there, except for the token `+` after `0+` and after `1+`, which carries mass of its own. In all, 250 of the 1,000 draws are valid, so $P(\mathcal{L}) = 1/4$ and rejection sampling needs 4 LM calls per valid sample. The other 750 are invalid, and the four kinds of invalid mass that Section 4 tells apart each appear: the root's bad tokens (300 draws), which every draw passes; the many light bad tokens at `0` and `1` (240), at nodes most draws pass; a single heavy atom, `+` after `0+` (100); and the bad tokens of the four depth-3 nodes (30), which the LM rarely reaches.

## 3 Speed: how much faster is CARS than ARS?

We compute the expected number of LM calls that each algorithm rejects before its $n$-th valid sample. We count in valid samples, not in calls, because the efficiency metric is calls per valid sample and because in this count the answer is exact.

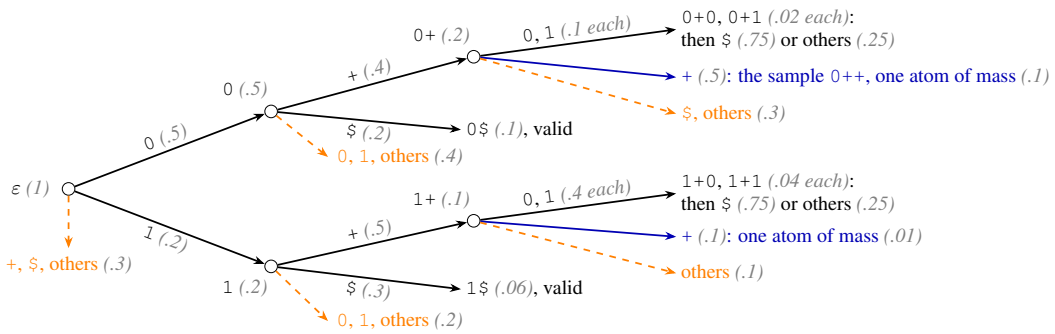
**Theorem 3.1** (Expected rejections)**.** *Write* $\mathrm{Rej}_S(n)$ *for the number of calls strategy* $S$ *rejects before its* $n$*-th valid sample, and let*

$$\rho_{v,a} = \frac{P(\mathcal{L})}{P(\mathcal{L}) + P(va)}, \qquad \rho_v = \frac{P(\mathcal{L} \setminus v\Sigma^*\$)}{P(\mathcal{L} \setminus v\Sigma^*\$) + P(v)}.$$

*Then for every* $n$*, with both sums over all atoms* $(v, a)$*,*

$$\mathbb{E}\,\mathrm{Rej}_{\mathrm{ARS}}(n) = \sum_{(v,a)} \big(1 - \rho_{v,a}^n\big), \qquad \mathbb{E}\,\mathrm{Rej}_{\mathrm{CARS}}(n) = \sum_{(v,a)} P(va \mid v)\,\big(1 - \rho_v^n\big).$$

Each component in the sums above is what an atom costs (being 1 for ARS) times the probability that the cost has been paid by the $n$-th valid sample. ARS removes the atom $(v, a)$ only by drawing it. The next sample that is either valid or in the atom is valid with probability $\rho_{v,a}$, so the atom survives $n$ valid samples with probability $\rho_{v,a}^n$, and removing it costs a whole rejected call. CARS removes all the atoms at $v$ with any draw through $v$, valid or not, so only the valid mass that avoids $v$ competes: the atoms survive $n$ valid samples with probability $\rho_v^n$, and removing them costs a




(a) The running example. Circles are grammatical prefixes $v$; tokens are in `typewriter` and probabilities in *(gray italics)*: $P(v)$ next to a node, $P(va \mid v)$ on an edge, and $P(v\$)$ next to a valid sequence. Black edges are valid continuations; orange dashed edges are invalid continuations, i.e., $\mathrm{atoms}(v)$, labelled with $P(v\mathrm{bad}_v \mid v)$; each orange edge stands for 100 light invalid tokens of equal mass. Blue edges are the two heavy invalid tokens.

| prefix $v$ | $P(v)$ | $P(v\mathrm{bad}_v \mid v)$ | atoms | $\rho_v$ | $\mathbb{E}\,\mathrm{Rej}_{\mathrm{CARS}}$ | $\mathbb{E}\,\mathrm{Rej}_{\mathrm{ARS}}$ |
|---|---|---|---|---|---|---|
| $\varepsilon$ | 1 | 0.30 | 100 | 0 | 0.30 | 69.7 |
| `0` | 0.50 | 0.40 | 100 | 0.19 | 0.40 | 54.9 |
| `1` | 0.20 | 0.20 | 100 | 0.39 | 0.20 | 14.8 |
| `0+` | 0.20 | 0.80 | 101 | 0.52 | 0.80 | 22.3 |
| `1+` | 0.10 | 0.20 | 101 | 0.66 | 0.20 | 4.9 |
| `0+0`, `0+1`, `1+0`, `1+1` | 0.02–0.04 | 0.25 | 100 each | 0.85–0.92 | 1.00 | 11.8 |
| total | | | 902 | | 2.9 | 178.4 |

(b) Theorem 3.1 on Example 2.1, prefix by prefix: the last two columns are the terms of the two sums of Theorem 3.1 for the atoms at $v$, i.e., the expected rejections each algorithm makes at $v$ before its 100th valid sample. For CARS that is $\sum_{a\in\mathrm{bad}_v} P(va \mid v)(1-\rho_v^{100})$, which is essentially $P(v\mathrm{bad}_v \mid v)$, paid at most once; for ARS it is $\sum_{a\in\mathrm{bad}_v}(1-\rho_{v,a}^{100})$, the expected number of the atoms at $v$ it has drawn, one rejection each. The last row sums the four depth-3 prefixes, each with $P(v\mathrm{bad}_v \mid v) = 0.25$ and 100 atoms.

Figure 1: The running example (Example 2.1) and the cost that Theorem 3.1 assigns to each prefix.

rejection only if the first visit to $v$ goes wrong, at $a$ with probability $P(va \mid v)$—summed over the atoms at $v$, at most $P(v\mathrm{bad}_v \mid v)$ per prefix. The proof is in Section A, and Section C gives the corresponding bounds when progress is counted in calls.

In Example 2.1, every sequence begins with $\varepsilon$, so $\rho_\varepsilon = 0$ and CARS removes the 100 root atoms with its first call. For ARS each root atom has mass 0.003, three draws in a thousand, and $\rho_{\varepsilon,a} = P(\mathcal{L})/(P(\mathcal{L}) + 0.003) = 0.988$, so it survives 100 valid samples with probability 0.30.

Figure 1b evaluates Theorem 3.1 on Example 2.1 at $n = 100$: CARS rejects 2.9 calls and ARS 178. At every prefix $\rho_v^{100}$ is negligible, so CARS has paid $P(v\mathrm{bad}_v \mid v)$ there and will pay nothing more. ARS has drawn 70 of the 100 root atoms and still pays for the rest.

**The whole run.** Letting $n$ grow bounds the rejections of an entire run.

**Corollary 3.2** (Lifetime rejections)**.** *Both expectations of Theorem 3.1 are nondecreasing in $n$, and*

$$\lim_{n\to\infty} \mathbb{E}\,\mathrm{Rej}_{\mathrm{ARS}}(n) = \big|\{(v,a) : P(va) > 0\}\big|, \qquad \lim_{n\to\infty} \mathbb{E}\,\mathrm{Rej}_{\mathrm{CARS}}(n) = \sum_{v:\,P(v)>0} P(v\mathrm{bad}_v \mid v).$$

In the limit, both algorithms remove every atom, and with them the same invalid mass $1 - P(\mathcal{L}) = \sum_{(v,a)} P(v)\,P(va \mid v)$. The two limits count the rejected calls each pays to get there, and they differ in what each atom costs. ARS removes an atom only by drawing it, and that draw is a rejection: it pays 1 per atom, however light, so over a run it makes every possible first mistake once. CARS removes all the atoms at $v$ with its first visit to $v$, and that visit is a rejection only if it goes wrong at $v$: it pays $P(va \mid v)$ per atom, at most one rejection per prefix. Because the expectations are nondecreasing in $n$, each is at most its limit. For CARS that limit is $\sum_v P(v\mathrm{bad}_v \mid v)$, a sum over

prefixes that does not depend on $n$, so $n$ valid samples cost CARS at most $n + \sum_v P(v\mathrm{bad}_v \mid v)$ calls in expectation, however large $n$ is. In the example the mass is $0.75$; ARS pays $902$ rejections to remove it and CARS $2.9$. The CARS total is also concentrated: whether CARS pays at $v$ depends only on the first token it draws there, so its lifetime rejections are a sum of independent coin flips, with variance at most the mean (Theorem A.3)—in the example $1.68$, against a mean of $2.9$. The ARS total is not: it counts how many distinct atoms the invalid draws hit, and every atom is exposed to the same invalid draws. How many of those occur before the $n$-th valid sample varies from run to run, and every atom's chance of being hit rises and falls with that one number, so the indicators are positively correlated (Theorem A.3(b)) and the total is more spread out than CARS's.

**No prefix-based strategy does better.** CARS removes, after every draw, all the mass the prefixes of that draw can reveal. On the stream coupling, therefore, any strategy that learns only from the prefixes of its draws—RS, ARS, or one that queries the oracle at only some prefixes—has excluded a subset of what CARS has at every point of the stream (Theorem A.4). This result strengthens the ordering CARS $\geq$ ARS $\geq$ RS of Parys et al. (2026) in three ways: it holds on every run rather than in expectation, against every prefix-based strategy at once, and for a reason—learning is monotone and costs nothing, so the rule that records everything the oracle says is the best one. In particular, when CARS gains little over ARS, no prefix-based strategy gains more. Prefix-based here means that after drawing $w$ the strategy removes only atoms at prefixes of $w$; a strategy that also queries the oracle at prefixes it has not drawn, or explores the trie on its own, is not covered by Theorem A.4.

## 4 Scope: when is CARS useful?

Section 3 counted progress in valid samples, $n$. This section counts it in LM calls, $k$—the *budget* a practitioner sets—because the question is now comparative: at the same budget, how much higher is the acceptance probability under CARS than under ARS? Write $\mathcal{W}_k^S$ for the state of strategy $S$ after $k$ calls; the next call is accepted with probability $P(\mathcal{L})/p_\varepsilon(\mathcal{W}_k^S)$. We define the factor by which ARS needs more calls per valid sample than CARS as follows:

$$\mathrm{Speedup}(k) = \frac{\mathbb{E}\, p_\varepsilon\big(\mathcal{W}_k^{\mathrm{ARS}}\big)}{\mathbb{E}\, p_\varepsilon\big(\mathcal{W}_k^{\mathrm{CARS}}\big)}.$$

It is at least 1 by the optimality of CARS (Section 3), and at most $1/P(\mathcal{L})$ because $P(\mathcal{L}) \leq p_\varepsilon \leq 1$. We ask which constraints and LMs make $\mathrm{Speedup}(k)$ large. The speedup comes from invalid mass that CARS has removed and ARS has not. CARS removes the atoms at $v$ once a call reaches $v$, and each call reaches $v$ with probability at least $P(v)$; after $k$ calls the atoms at $v$ remain with probability at most $(1-P(v))^k$. ARS removes an atom of mass $m$ only by drawing it, and each call draws it with probability at most $m/P(\mathcal{L})$; after $k$ calls the atom remains with probability at least $(1 - m/P(\mathcal{L}))^k$ (Theorem A.2). So within $k$ calls CARS reaches the prefixes of probability well above $1/k$, and ARS misses the atoms of mass well below $P(\mathcal{L})/k$.

**Theorem 4.1** (Scope)**.** *For thresholds $t$ and $s$, let*

$$R_t = \sum_{v:\, P(v)\geq t} P(v)\, P(v\mathrm{bad}_v \mid v), \qquad U_t = 1 - P(\mathcal{L}) - R_t, \qquad \Lambda_s = \sum_{(v,a):\, P(va)<s} P(va).$$

*$R_t$ is the* reachable *invalid mass at prefixes of probability at least $t$; $U_t$ is the* unreachable *rest; and $\Lambda_s$ is the light mass in atoms $<s$. For every number of calls $k$, every $t$, and every $s \leq P(\mathcal{L})$,*

$$\mathrm{Speedup}(k) \;\geq\; \frac{P(\mathcal{L}) + \big(1 - s/P(\mathcal{L})\big)^k\, \Lambda_s}{P(\mathcal{L}) + U_t + (1-t)^k\, R_t} \;\Bigg\} \;\leq\; \frac{\mathbb{E}\, p_\varepsilon(\mathcal{W}_k^{\mathrm{ARS}})}{\mathbb{E}\, p_\varepsilon(\mathcal{W}_k^{\mathrm{CARS}})}.$$

Both bounds follow from the two survival bounds above by summing over atoms.

The one restriction, $s \leq P(\mathcal{L})$, is where ARS's power means something: the survival bound $(1 - m/P(\mathcal{L}))^k$ above is a bound only while $m \leq P(\mathcal{L})$. An atom heavier than the valid mass $P(\mathcal{L})$ is never light for ARS, at any budget $k$: by Theorem 3.1 the atom survives even the first valid sample with probability $\rho_{v,a} = P(\mathcal{L})/(P(\mathcal{L}) + m) < 1/2$. Each threshold is a trade-off. Raising $t$ shrinks $(1-t)^k$, the bound on the chance that CARS has not yet reached a prefix counted in $R_t$, but moves

prefixes from $R_t$ into $U_t$, which CARS is charged for in full; lowering $s$ brings $(1 - s/P(\mathcal{L}))^k$ closer to 1, but leaves fewer atoms in $\Lambda_s$. Where the balance lies depends on the budget $k$, and this is how $k$ enters the choice. With $t = c/k$ and $s = \delta P(\mathcal{L})/k$, we have $(1-t)^k \leq e^{-c}$ and $(1 - s/P(\mathcal{L}))^k \geq 1 - \delta$, so

$$\text{Speedup}(k) \;\geq\; \frac{P(\mathcal{L}) + (1-\delta)\,\Lambda_{\delta P(\mathcal{L})/k}}{P(\mathcal{L}) + U_{c/k} + e^{-c} R_{c/k}}.$$

A few multiples of $1/k$ make CARS's leftover small ($e^{-3} = 0.05$), and a small fraction of $P(\mathcal{L})/k$ makes ARS's factor close to 1. After $k$ calls, two masses decide the speedup, one per algorithm. Write $\Lambda^{(k)}$ for the light mass at $s \approx P(\mathcal{L})/k$: the invalid mass in atoms so light that ARS has most likely not drawn them yet, since each call draws such an atom with probability at most about $1/k$. Write $U^{(k)}$ for the unreachable mass at $t \approx 1/k$: the invalid mass at prefixes so rare that CARS has most likely not reached them yet, since each call reaches them with probability below about $1/k$. ARS's proposal still holds at least $P(\mathcal{L}) + \Lambda^{(k)}$, and CARS's at most $P(\mathcal{L}) + U^{(k)}$:

$$\text{Speedup}(k) \;\gtrsim\; \frac{P(\mathcal{L}) + \Lambda^{(k)}}{P(\mathcal{L}) + U^{(k)}}.$$

When $U^{(k)} \approx 0$ and $\Lambda^{(k)} \approx 1 - P(\mathcal{L})$, the speedup is the ceiling $1/P(\mathcal{L})$. When $U^{(k)} \approx 1 - P(\mathcal{L})$, CARS reaches nothing within the budget and the speedup is close to 1. Both masses are read in units of $P(\mathcal{L})$: mass $m$ that stays in the proposal costs $m/P(\mathcal{L})$ extra calls per valid sample.

In Example 2.1 at $k = 20$, with the thresholds at $1/k$ and $P(\mathcal{L})/k$ (i.e., $c = \delta = 1$), the prefixes of probability below $1/k = 0.05$ are exactly the four depth-3 prefixes, so $U^{(20)} = U_{0.05} = 0.03$. The only atom heavier than $P(\mathcal{L})/k = 0.0125$ is + after `0+`, so $\Lambda^{(20)} = \Lambda_{0.0125} = 0.65$. The estimate is $(0.25+0.65)/(0.25+0.03) = 3.2$, against a simulated speedup of 3.5 and the ceiling 4 (Section D); Theorem 4.1, maximized over $t$ and $s$, gives 2.4. Section B sharpens both powers, using that $p_\varepsilon$ falls during a run, and refines the thresholds into histograms; Section 5 uses that form.

**Kinds of constraints.** The masses $\Lambda^{(k)}$ and $U^{(k)}$ sort constraints into four kinds, one for each kind of invalid mass in Example 2.1; Section E sorts the benchmarks of Parys et al. (2026) into them.

*Format constraints*, such as a mandatory keyword or wrapper, put the invalid mass at the root or along a *spine*, a chain of prefixes that every valid sequence begins with. A spine prefix has $\rho_v = 0$, so CARS clears the spine within its first few calls, however small $P(\mathcal{L})$ is; ARS must draw each of its bad tokens, up to a vocabulary's worth, one rejection apiece (e.g., the root in Figure 1b).

*Tight per-position grammars*, which admit few tokens at each position, put many light bad tokens at every prefix. ARS cannot draw them within the budget, so it keeps spending calls on them, $\Lambda^{(k)}/P(\mathcal{L})$ extra per valid sample: $\Lambda^{(k)}$ is large, and so is the speedup. In Figure 1b, these are the bad tokens at `0` and `1`, 240 draws in a thousand spread over 200 tokens.

*Loose grammars/blacklists*, with few bad tokens per position, make each bad token heavy. ARS draws a heavy atom within a few valid samples, so little of the mass is light, $\Lambda^{(k)}$ is small, and CARS leads by a small factor. In Figure 1b, this is + after `0+`, drawn after $\sim 2.5$ valid samples.

*Global constraints*, such as "at least three key–value pairs", put the invalid mass at rare prefixes. $U^{(k)}$ is large and no prefix-based strategy reaches that mass within the budget. In Figure 1b, these are the four depth-3 prefixes. Exact methods are not a good fit for such constraints.

## 5 The predictions against the benchmarks

We test Theorems 3.1 and 4.1 on the benchmarks of Parys et al. (2026). We ask: (i) does the trie of one CARS run predict the cost of independent runs? (ii) do the masses $\Lambda^{(k)}$ and $U^{(k)}$ explain where CARS is fast? (iii) how early in a run can the cost of the whole run be predicted?

We use nine tasks with the grammars and prompts of the CARS repository—SMILES (acrylates) (Weininger, 1988; Guo et al., 2022), SyGuS SLIA and BV4 (Alur et al., 2019), and the JSON, XML and SQL fuzzing grammars of Parys et al. (2026), the last three also with the grammar in the prompt—on Llama-3.1-8B-Instruct at temperature 1 without truncation, as the analysis assumes.

| | | RS | | ARS | | CARS | |
|---|---|---|---|---|---|---|---|
| Benchmark | $\hat{P}(\mathcal{L})$ | measured | predicted | measured | predicted | measured | predicted |
| SMILES (acrylates) | 0.049 | 2,109 | 2,055 | 776 | 560–803 | 489 | 271–515 |
| SyGuS SLIA | $8.5 \times 10^{-4}$ | 6 / 8,000* | $1.2 \times 10^{5}$ | 991 | 876–883 | 115 | 111–119 |
| SyGuS BV4 | $2.7 \times 10^{-5}$ | 0 / 8,000* | $3.7 \times 10^{6}$ | 3,341 | 3,610–3,666 | 177 | 139–195 |
| JSON | $6.5 \times 10^{-7}$ | – | $1.5 \times 10^{8}$ | 29 / 8,000* | 15,420–16,174 | 700 | 105–858 |

Table 1: Measured and predicted LM calls for 100 valid samples on four tasks; full table in Table 2 (Section F). Predictions are Theorem 3.1 from the trie of the seed-0 CARS run. The lower end of a range is Theorem 3.1 on the atoms the trie saw; the upper end adds the unseen invalid mass at the rejection-sampling rate (that mass is known only to within the `float32` precision of the trie's masses, $\sim 10^{-6}$. *: budget exhausted before 100 valid samples; valid / calls.

CARS runs ten times and ARS (due to its common timeouts and the fact that we are mostly interested in CARS), each to 100 valid samples or a budget (Section G). Predictions use the trie of one CARS run, what a practitioner has after running CARS once, so the other runs test them: the trie gives every prefix and atom mass exactly, and the esitmate $\hat{P}(\mathcal{L})$ comes from its 100 accepted samples. Table 9 in Section F repeats the comparison on seven more models.

**The trie predicts the cost.** How sharp the prediction is depends on how much of the invalid mass the run has visited (Table 1). On SMILES much of it lies at prefixes the run never reached, so the prediction is a range and the measured cost sits in its upper part. On SyGuS SLIA and BV4 almost nothing is unseen, and the measured CARS cost lies inside the predicted range. Rejection sampling needs no trie at all—its cost is $100/\hat{P}(\mathcal{L})$—and it checks that constant on its own: at the same budget of 8,000 calls it took 2,109 calls for 100 valid samples on SMILES against the predicted 2,055, and produced 6 and 0 valid samples on SLIA and BV4, where 100 would cost a predicted $1.2 \times 10^5$ and $3.7 \times 10^6$ calls. The LM almost never begins with `(define-fun`: the root carries $P(\varepsilon\mathrm{bad}_\varepsilon \mid \varepsilon) = 0.98$ (SLIA) and 1.00 (BV4), and, as Corollary 3.2 says, CARS pays for the root at most once, on its first call, while ARS must draw its 128,255 bad tokens one at a time. On JSON the measured cost again sits in the upper part of the range: at the end of the run $p_\varepsilon$ is still 7 times $P(\mathcal{L})$, and every valid sample pays for that unseen mass at the rejection-sampling rate. The five remaining tasks—XML, SQL and the three grammar-in-prompt variants—are in Table 2; on SQL and with the grammar in the prompt, $\hat{P}(\mathcal{L})$ falls below the precision to which the trie's masses are known, so the predictions there are lower bounds, and CARS still reaches 100 valid samples on every one of them, on SQL in 2,901 calls where ARS finds none in 8,000 (Section F).

**Light mass $\Lambda_s$ and unreachable mass $U_t$ explain the speedup.** Theorem 4.1 explains the speedup by two masses read off the same trie: $\Lambda^{(k)}$, the invalid mass in atoms too light for ARS to draw within $k$ calls, and $U^{(k)}$, the invalid mass at prefixes too rare for CARS to reach. We evaluate the bound at the thresholds $(t^*, s^*)$ that make the speedup largest on the seed-0 trie. What the bound resolves depends on where the invalid mass sits. Call the root *bad* when the LM almost never begins an output validly, $P(\varepsilon\mathrm{bad}_\varepsilon \mid \varepsilon) \approx 1$: a format constraint in the sense of Section 4, and the case on every task here but SMILES (0.98 on SLIA, 1.00 on the rest). There the bound tracks the measured speedup—12 against 15 on SLIA, 55 against 98 on BV4, 247 against 279 on JSON and 17 against 19 on XML, within a factor of 1.1–1.8 (Table 4 in Section F)—and the two masses say why. $U_{t^*}/\hat{P}(\mathcal{L})$ stays at a few calls per valid sample (1–6) while $\Lambda_{s^*}/\hat{P}(\mathcal{L})$ rises with the measured speedup, from 12 on SLIA to 1,380 on JSON: the speedup is carried by light invalid mass that ARS does not draw within the budget, as Theorem 4.1 says. SMILES is the task the bound does not resolve: $P(\varepsilon\mathrm{bad}_\varepsilon \mid \varepsilon) = 0.001$ and invalid mass sits in atoms ARS collects after a few valid samples, so $U_{t^*}$ exceeds $\Lambda_{s^*}$ and the bound returns 1 where the truth is 1.8, a fair verdict at this budget.

**Larger models: the gain follows the invalid mass, not the size.** Parys et al. (2026) report that the advantage of CARS over ARS grows with the size of the model. The analysis says that what matters is not size but where the model puts its invalid mass, and Llama-3.1-70B, run on SMILES and SLIA against the 8B model, moves it in opposite directions on the two. On SMILES the larger model is better aligned with the grammar: $\hat{P}(\mathcal{L})$ rises from 0.049 to 0.28, the ceiling $1/P(\mathcal{L})$ falls

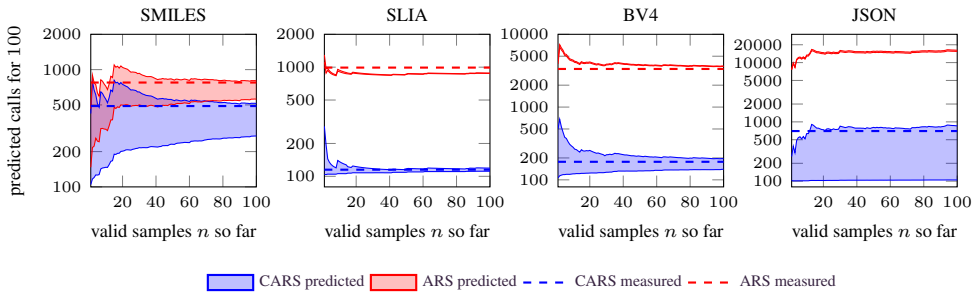



Figure 2: Predicted LM calls for 100 valid samples, from the seed-0 CARS run's trie after its $n$-th valid sample (shaded: the range of Table 1), against the measured mean over all runs (dashed); no ARS run reached 100 valid samples on JSON.

from 21 to 3.5, and the ratio of ARS to CARS calls for 100 valid samples falls from 1.6 to 1.2. On SLIA it is further from the SyGuS syntax: $\hat{P}(\mathcal{L})$ falls from $8.5 \times 10^{-4}$ to $1.3 \times 10^{-5}$, and the ratio rises from 8.6 to 68 (101 calls against 6,887), where Theorem 3.1 predicts 66. A larger model gains more from CARS only when it puts more of its mistakes where CARS removes them cheaply.

**A run predicts its cost in advance.** Figure 2 asks how early the prediction arrives, analyzing the trie as it stands after each sample, with $P(\mathcal{L})$ estimated from the calls so far, and using the analysis to predict the cost of producing 100 valid samples. For CARS the measured cost comes within 5% of the predicted range, and stays there, from the 4th call on SLIA, the 7th on BV4, the 19th on SMILES and the 69th on JSON—3–10% of the 115–661 calls the run takes—because the cost is a sum over the prefixes holding invalid mass (Theorem 3.1), which a bad root puts on the first few calls. ARS settles later, within 30% after 40 calls and within 12% after 100 valid samples on SyGuS, since its cost rests on $\hat{P}(\mathcal{L})$, whose relative error is about $1/\sqrt{\text{accepted samples}}$; the $\text{Speedup}(100)$ bound is within 6% of its final value after ten valid samples on all four tasks. A few percent of a CARS job thus tell what CARS will cost and what it saves, and how far to trust that: the range's width is the unseen invalid mass in units of $\hat{P}(\mathcal{L})$ (Section F).

## 6 Related Work

This paper analyzes the algorithms of Parys et al. (2026). CARS descends from adaptive rejection sampling in the sense of Mansinghka et al. (2009), which removes each rejected output from the proposal, and from the adaptive rejection sampler of Gilks & Wild (1992), which tightens an envelope of a log-concave density at the points where it is evaluated; neither line analyzes the rate at which the adapted proposal improves (Sheldon, 2013; Devroye, 2012). Theorem 3.1 gives that rate exactly, and its ARS statement—one rejection per possible mistake—is, to our knowledge, the first closed form for the cost of an ARS-type sampler. The stream coupling is the standard view of rejection sampling as thinning; what is new is that under CARS the thinned-out samples carry no information (Lemma A.1). Tromble & Eisner (2006) patch a model with constraints as violations are discovered, for argmax decoding over weighted automata; their setting has no sampling cost.

Grammar-constrained decoding (Scholak et al., 2021; Beurer-Kellner et al., 2023; Geng et al., 2023; Melcer et al., 2024; Park et al., 2025) masks tokens that cannot be completed validly and never rejects, at the price of distorting the LM's conditional distribution (Park et al., 2024; Tam et al., 2024); it implements the prefix oracle CARS queries, and our budget bound applies to any method learning only through such an oracle. Asymptotically exact methods—ASAp (Park et al., 2024), Markov chain and sequential Monte Carlo (Anaya Gonzalez et al., 2025; Lew et al., 2023; Lipkin et al., 2025)—converge only in the limit, and their analyses bound distance to the target in steps or particles; for the exact methods studied here the distribution is correct from the first sample and only the cost is in question. The closest analogue there is the choice of step or particle count, for which the CARS paper notes there is no principled stopping rule; Theorem 3.1 and Corollary 3.2 instead bound a run's total cost in advance. Surrogate- (Zhang et al., 2023; 2024) and gradient-based (Qin et al., 2022; Amini et al., 2023; Kumar et al., 2022) control do not sample exactly.

## Reproducibility Statement

Part of the analysis is machine-checked in Lean 4 with Mathlib, and Section H says exactly which part, theorem by theorem: the per-atom survival bounds, the identity expressing $\mathbb{E}[p_\varepsilon]$ as a sum over atoms, the two threshold bounds behind Theorem 4.1 together with the envelopes they use, the exact per-atom rates, exactness and monotonicity of the samplers, and the consequences of Theorem A.4 are checked. The stream coupling of Section 2 (Lemma A.1) and the results proved through it—Theorem 3.1, Theorem A.3 and Corollaries C.1 to C.2—are proved on paper only. The obstacle is the shape of the formalization rather than the difficulty of the proofs: the Lean model defines a strategy's cost by recursion on the number of draws, and that recursion never names a probability space, so it cannot state the independence of the per-prefix indicators that Theorem A.3(a) asserts. Section G identifies every checkpoint by its Hugging Face revision, with the library versions, the GPUs used and the commit of the CARS implementation the harness runs on. Section F states what was run: the tasks and prompts, ten CARS and five ARS runs per task with five RS runs on the three tasks where rejection sampling can finish, the per-strategy call budgets, and which single run supplies the trie that every prediction is computed from, so that the remaining runs test it. It also states how the per-prefix quantities of Theorem 3.1 are read off a run rather than estimated, and how the invalid mass a run never reached is bracketed. The harness was validated end to end on a CPU-sized model over thirty independent trials before the benchmark runs (Section G).

## AI Use Statement

In this work, we used generative AI tools (large language models) for the following tasks with required disclosure: formulating mathematical claims and providing ingredients of their proofs, including finding the improved bounds on the cost of CARS; assisting in the writing of proofs, both the proofs in the paper and the Lean 4 formalization of Section H, which LLMs wrote; refining research hypotheses; designing and giving feedback on the experiments; implementing the evaluation code and running the experiments; and interpreting the experimental results. We have not used generative AI tools to generate synthetic data sets, and translation, dataset cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, produce the tables and figures from the experimental data, draft and edit parts of the paper, and search for, verify and format references. We have reviewed all AI-assisted work. Every formal proof is checked by the Lean kernel, so the correctness of the formalized results does not depend on the model that produced them; the paper proofs were checked by the authors; the numbers of the running example were checked by exact enumeration in rational arithmetic; and every reference was checked against its publisher or arXiv record. The authors directed the work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## References

Guidance AI. llguidance: Super-fast structured outputs, 2025. URL https://github.com/guidance-ai/llguidance. Accessed: 2025-09-23.

Rajeev Alur, Dana Fisman, Saswat Padhi, Rishabh Singh, and Abhishek Udupa. Sygus-comp 2018: Results and analysis, 2019.

Afra Amini, Li Du, and Ryan Cotterell. Structured Voronoi sampling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 31689–31716. Curran Associates, Inc., 2023. doi: 10.52202/075280-1376. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/64ae05e3f1a88ebac7f9263b69f4e702-Paper-Conference.pdf.

Emmanuel Anaya Gonzalez, Sairam Vaidya, Kanghee Park, Ruyi Ji, Taylor Berg-Kirkpatrick, and Loris D'Antoni. Constrained sampling for language models should be easy: An MCMC perspective. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 59951–59981. Curran Associates, Inc., 2025. doi: 10.52202/

085713-2005. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5681251fa039cf49d6d11b906eded1b3-Paper-Conference.pdf.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI), 2023. doi: 10.1145/3591300.

Luc Devroye. A note on generating random variables with log-concave densities. *Statistics & Probability Letters*, 82(5):1035–1039, 2012. doi: 10.1016/j.spl.2012.01.022.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.674. URL https://aclanthology.org/2023.emnlp-main.674/.

W. R. Gilks and P. Wild. Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):337–348, 1992. doi: 10.2307/2347565.

Minghao Guo, Veronika Thost, Beichen Li, Payel Das, Jie Chen, and Wojciech Matusik. Data-efficient graph grammar learning for molecular generation, 2022. URL https://arxiv.org/abs/2203.08031.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. Gradient-based constrained sampling from language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2251–2277, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.144. URL https://aclanthology.org/2022.emnlp-main.144/.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K. Mansinghka. Sequential Monte Carlo steering of large language models using probabilistic programs. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023. URL https://openreview.net/forum?id=Ul2K0qXxXy.

Benjamin Lipkin, Benjamin LeBrun, Jacob Hoover Vigly, João Loula, David R. MacIver, Li Du, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Timothy J. O'Donnell, Alexander K. Lew, and Tim Vieira. Fast controlled generation from language models with adaptive weighted rejection sampling. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2504.05410.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexander Lew, Tim Vieira, and Timothy O'Donnell. Syntactic and semantic control of large language models via sequential Monte Carlo. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, pp. 64758–64791, 2025. URL https://openreview.net/forum?id=xoXn62FzD0.

Vikash Mansinghka, Daniel Roy, Eric Jonas, and Joshua Tenenbaum. Exact and approximate sampling by systematic stochastic search. In David van Dyk and Max Welling (eds.), *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 400–407, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL https://proceedings.mlr.press/v5/mansinghka09a.html.

Daniel Melcer, Nathan Fulton, Sanjay Krishna Gouda, and Haifeng Qian. Constrained decoding for fill-in-the-middle code language models via efficient left and right quotienting of context-sensitive grammars, 2024. URL https://arxiv.org/abs/2402.17988. arXiv:2402.17988.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 24547–24568. Curran Associates, Inc., 2024. doi: 10.52202/079017-0774. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2bdc2267c3d7d01523e2e17ac0a754f3-Paper-Conference.pdf.

Kanghee Park, Timothy Zhou, and Loris D'Antoni. Flexible and efficient grammar-constrained decoding. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48262–48275. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/park25l.html.

Paweł Parys, Sairam Vaidya, Taylor Berg-Kirkpatrick, and Loris D'Antoni. Constrained adaptive rejection sampling. In *International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=gLRqqQOM9K.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. COLD decoding: Energy-based constrained text generation with langevin dynamics. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 9538–9551. Curran Associates, Inc., 2022. doi: 10.52202/068431-0693. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3e25d1aff47964c8409fd5c8dc0438d7-Paper-Conference.pdf.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Daniel R. Sheldon. Discrete adaptive rejection sampling. Technical report, University of Massachusetts Amherst, 2013. URL https://people.cs.umass.edu/~sheldon/papers/discrete_ars.pdf.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? a study on the impact of format restrictions on large language model performance. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1218–1236, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.91. URL https://aclanthology.org/2024.emnlp-industry.91/.

Roy Tromble and Jason Eisner. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In Robert C. Moore, Jeff Bilmes, Jennifer Chu-Carroll, and Mark Sanderson (eds.), *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 423–430, New York City, USA, June 2006. Association for Computational Linguistics. URL https://aclanthology.org/N06-1054/.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. Syncode: Llm generation with grammar augmentation, 2024. URL https://arxiv.org/abs/2403.01632.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van Den Broeck. Tractable control for autoregressive language generation. 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. 40932–40945. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/zhang23g.html.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. Adaptable logical control for large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 115563–115587. Curran Associates, Inc., 2024. doi: 10.52202/079017-3670. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d15c16cf5619a2b1606da5fc88e3f1a9-Paper-Conference.pdf.

# A PROOFS

The proofs, and their Lean counterparts in Section H, are written once for a set of atoms $A$ and its *discovery set* $D$, the draws that remove $A$, and read once per strategy (a set is *pruned* once it has been added to the excluded set $\mathcal{W}$):

$$A = A_{v,a}, \qquad D_{v,a} = A_{v,a} \qquad \text{(ARS: an atom is discovered by drawing it)},$$
$$A = \mathrm{atoms}(v), \qquad D_v = v\Sigma^*\$ \qquad \text{(CARS: by drawing anything through } v\text{)},$$

with $P(D_{v,a}) = P(va)$ and $P(D_v) = P(v)$.

**Lemma A.1** (Skipped samples are inert)**.** *After any number $N$ of stream samples $w_1, \ldots, w_N$, CARS has removed exactly the atoms at the nodes $\bigcup_{i \le N} \mathrm{path}(w_i)$, and ARS exactly the atoms containing some $w_i$, $i \le N$—whether or not those samples were skipped.*

*Proof.* A sample skipped by CARS lies in $\mathrm{atoms}(u)$ for a pruned node $u$, so it makes its first mistake at $u$ and its path is $u$ with its ancestors; all were pruned by the earlier draw through $u$. A sample skipped by ARS lies in a removed atom. An LM call removes its path (CARS) or its atom (ARS). □

**Theorem A.2** (One node)**.** *Assume $P(\mathcal{L}) > 0$, and $P(v) > 0$ wherever a node $v$ is named.*

*(a)* (CARS: one visit clears the bad tokens.) *After $k$ LM calls, $\mathrm{atoms}(v)$ is still not entirely pruned with probability at most $(1 - P(v))^k$. The same bound holds simultaneously for the bad tokens of every ancestor of $v$, since a draw through $v$ passes through each of them and so clears them all at once.*

*(b)* (CARS, exactly.) *$\mathrm{atoms}(v)$ is still not pruned when the $n$-th valid sample is produced with probability exactly $\rho_v^n$, and the number of valid samples CARS produces before it clears $v$ is geometric with mean $P(\mathcal{L} \setminus (v\Sigma^*\$))/P(v)$.*

*(c)* (ARS: one rejection clears one atom.) *Along every run, if $n$ nonempty atoms have been pruned after $k$ LM calls then $k \ge n$, so clearing $\mathrm{atoms}(v)$ costs ARS at least $|\mathrm{bad}_v|$ rejected draws. An atom of mass $m \le P(\mathcal{L})$ is still unpruned after $k$ calls with probability at least $(1 - m/P(\mathcal{L}))^k$ and, exactly, the atom $(v, a)$ is still unpruned when the $n$-th valid sample is produced with probability $\rho_{v,a}^n = \big(P(\mathcal{L})/(P(\mathcal{L}) + P(va))\big)^n$: ARS produces $P(\mathcal{L})/P(va)$ valid samples on average before it stumbles on that atom.*

**Theorem A.2.** Part (a) uses that draws outside $D_v$ never exclude anything inside $D_v$: a draw not through $v$ adds invalid children of its own prefixes, none of which is comparable with $v$. So until the first visit to $v$ the whole mass $P(v)$ is available, and each call is a draw from $P$ with the excluded set removed and renormalized by $p_\varepsilon(\mathcal{W}) \le 1$ (the sampling step of Parys et al. (2026), Theorem 3.2 there), so it reaches $v$ with probability at least $P(v)$; an induction on $k$ with the invariant $\mathcal{W}\Sigma^*\$ \cap D_v = \emptyset$ gives the bound (`surv_le_pow_of_disjoint`). Part (c)'s first claim is combinatorial: under ARS the excluded set after draws $w_1, \ldots, w_k$ is the union of the atoms containing the $w_i$, and atoms are disjoint. The exact statements are a race. Let $A$ be the set in question with discovery set $D$ ($\mathrm{atoms}(v)$ and $D_v = v\Sigma^*\$$ for CARS, $A_{v,a}$ and $D_{v,a} = A_{v,a}$ for ARS), and call a stream sample *relevant* if it lies in $(\mathcal{L} \setminus D) \cup D$; every valid sample is relevant, and an irrelevant sample neither counts as valid nor, by Lemma A.1, prunes $A$. $A$ survives the $n$-th valid sample iff the first

$n$ relevant samples all lie in $\mathcal{L} \setminus D$, which has probability $\big(P(\mathcal{L} \setminus D)/(P(\mathcal{L} \setminus D) + P(D))\big)^n$; for ARS $\mathcal{L} \setminus A_{v,a} = \mathcal{L}$.

**Theorem 3.1.** *CARS.* A rejection is an invalid LM call $w$; let $u$ be its *first-mistake node*, the grammatical prefix $v$ of its shortest invalid prefix $va$. Since $w$ was not skipped, $u$ is unpruned when $w$ is drawn, and $w$ prunes it. So distinct rejections have distinct first-mistake nodes, and $\mathrm{Rej}_{\mathrm{CARS}}(n)$ counts the nodes $u$ whose first visit happens before $T_n$ and is by a sample that makes its first mistake at $u$. With the relevant samples of $u$ as in the proof of Theorem A.2, this is the event that the first sample through $u$ in the relevant subsequence lies in $\mathrm{atoms}(u)$ and is preceded by fewer than $n$ samples of $\mathcal{L} \setminus (u\Sigma^*\$)$:

$$\sum_{j=0}^{n-1} \rho_u^j \cdot \frac{P(u)P(u\mathrm{bad}_u \mid u)}{P(\mathcal{L} \setminus (u\Sigma^*\$)) + P(u)} = \frac{1-\rho_u^n}{1-\rho_u} \cdot (1-\rho_u)\, P(u\mathrm{bad}_u \mid u) = P(u\mathrm{bad}_u \mid u)\big(1-\rho_u^n\big).$$

*ARS.* A rejection is a call into an unpruned atom, and it prunes that atom, so $\mathrm{Rej}_{\mathrm{ARS}}(n)$ is the number of atoms hit before $T_n$; use Theorem A.2(c). *RS.* A sum of $n$ independent geometrics, one per valid sample (negative binomial), with mean $n(1-P(\mathcal{L}))/P(\mathcal{L})$.

**Theorem A.3** (Distribution of the waste)**.** *Let $\mathrm{Rej}_S(\infty) = \lim_n \mathrm{Rej}_S(n)$ be the number of rejected LM calls over an entire run, and assume finitely many grammatical prefixes (as under a length cap).*

*(a)* (CARS.) $\mathrm{Rej}_{\mathrm{CARS}}(\infty) = \sum_v B_v$*, where the $B_v$ are* independent *Bernoulli variables with $\mathbb{P}[B_v = 1] = P(v\mathrm{bad}_v \mid v)$, one per frontier node with $P(v) > 0$. Hence, with $\mu = \sum_v P(v\mathrm{bad}_v \mid v)$,*

$$\mathrm{Var}\big[\mathrm{Rej}_{\mathrm{CARS}}(\infty)\big] = \sum_v P(v\mathrm{bad}_v \mid v)\big(1 - P(v\mathrm{bad}_v \mid v)\big) \le \mu,$$

$$\mathbb{P}\big[\mathrm{Rej}_{\mathrm{CARS}}(\infty) \ge \mu + t\big] \le \exp\Big(-\frac{t^2}{2(\mu + t/3)}\Big) \qquad (t \ge 0),$$

*and since $\mathrm{Rej}_{\mathrm{CARS}}(n) \le \mathrm{Rej}_{\mathrm{CARS}}(\infty)$ on every run, the tail bound holds for $\mathrm{Rej}_{\mathrm{CARS}}(n)$ at every $n$.*

*(b)* (ARS.) *For every $n$, $\mathrm{Var}[\mathrm{Rej}_{\mathrm{ARS}}(n)]$ is the independent-indicator value $\sum_{(v,a)} \rho_{v,a}^n(1-\rho_{v,a}^n)$ plus a nonnegative covariance for every pair of atoms, in closed form below.*

The two parts differ in kind. CARS's waste is a sum of independent coin flips, one per node, so it concentrates. ARS's is an occupancy count—how many distinct atoms the invalid draws hit—and the number of invalid draws before $T_n$ is shared by all atoms, which makes the covariances positive (threefold in the example, Section D).

**Theorem A.3.** Part (b) in full: with $f_{v,a} = \rho_{v,a}$ and $h_{(v,a),(v',b)} = P(\mathcal{L})/(P(\mathcal{L}) + P(va) + P(v'b))$,

$$\mathrm{Var}\big[\mathrm{Rej}_{\mathrm{ARS}}(n)\big] = \sum_{(v,a)} f_{v,a}^n\big(1 - f_{v,a}^n\big) + \sum_{(v,a)\neq(v',b)} \Big(h_{(v,a),(v',b)}^n - f_{v,a}^n f_{v',b}^n\Big),$$

and every term of the second sum is nonnegative.

(a) An autoregressive LM produces a sample by drawing, at each prefix it reaches, a next token from $P(\cdot \mid u)$ independently of everything drawn before; so the i.i.d. stream can be generated by attaching to every prefix $u$ an i.i.d. sequence of tokens from $P(\cdot \mid u)$, independent across prefixes, and letting each sample consume the next unused token at every prefix on its path. Let $B_v = 1$ iff the first token drawn at $v$ lies in $\mathrm{bad}_v$. A rejection of CARS with first-mistake node $v$ is the first sample through $v$ (else $v$ would be pruned and the sample skipped), and its token at $v$ is bad; conversely, if the first token drawn at $v$ is bad, the first sample through $v$ makes its first mistake at $v$, lies in $\mathrm{atoms}(v)$ with $v$ unpruned, and is therefore an LM call that is rejected. So $\mathrm{Rej}_{\mathrm{CARS}}(\infty) = \sum_v B_v$, and $B_v$ is a function of the token sequence attached to $v$ alone. The variance is that of independent indicators, and the tail bound is Bernstein's inequality for a sum of independent $[0,1]$-valued variables with variance at most $\mu$.

(b) $\mathrm{Rej}_{\mathrm{ARS}}(n) = \sum_{(v,a)} \mathbf{1}[A_{v,a} \text{ hit before } T_n]$. By Theorem A.2(c) each indicator has mean $1 - f^n_{v,a}$. For two atoms, apply the race of Theorem A.2 with $D = A_{v,a} \cup A_{v',b}$: neither is hit by $T_n$ iff the first $n$ relevant samples are valid, which has probability $h^n$ with $h = P(\mathcal{L})/(P(\mathcal{L}) + P(va) + P(v'b))$. Hence $\mathbb{P}[\text{both hit}] = 1 - f^n_{v,a} - f^n_{v',b} + h^n$ and the covariance is $h^n - f^n_{v,a} f^n_{v',b}$; this is nonnegative because $h \geq f_{v,a} f_{v',b}$, as $(P(\mathcal{L}) + P(va))(P(\mathcal{L}) + P(v'b)) \geq P(\mathcal{L})(P(\mathcal{L}) + P(va) + P(v'b))$.

**What Theorem A.4 covers.** A *strategy* $S$ here is any rule that, after each LM call $w$, adds sequences to its excluded set $\mathcal{W}$ and never excludes anything valid; it proposes from $P$ with $\mathcal{W}$ removed, exactly as CARS does, and differs from CARS only in what it learns from $w$. It is *prefix-based* if what it learns from $w$ lies on the path of $w$: after drawing $w$ it removes some subset of $\bigcup_{u \in \mathrm{path}(w)} \mathrm{atoms}(u)$, the bad tokens at the nodes $w$ passed through. This covers rejection sampling (removes nothing), ARS (removes the atom containing $w$), CARS (removes everything the oracle reports along $w$) and everything between them: an oracle consulted only at some positions or only up to some depth, a strategy that removes the bad tokens at the first-mistake node alone, or one that forgets. It excludes strategies that learn about prefixes they have not drawn—querying the oracle at siblings of the path, or enumerating heavy prefixes offline as in Section G—and anything that changes the proposal other than by exclusion, such as reweighting or resampling. Within the class the comparison is pathwise: all strategies read the same stream, so the proposition speaks of every run, not of expectations. The Lean form (`Dominates`, Section H) is the abstract version: $S_2$ dominates $S_1$ when every draw teaches $S_2$ at least what it teaches $S_1$ and $S_2$'s excluded sets are closed under $S_1$'s updates; with CARS as $S_2$ this is exactly the prefix-based condition.

**Theorem A.4** (Optimality)**.** *Let $S$ be a strategy that, after drawing $w$, removes only atoms at prefixes of $w$. Run $S$ and CARS on the same stream. Then at every point every sequence removed by $S$ is also removed by CARS, so before the $n$-th valid sample every call CARS rejects is also rejected by $S$, and among its first $k$ calls CARS rejects no more than $S$ does.*

*Proof.* By induction on $N$: $S$'s LM calls are stream samples, and $S$ adds a subset of $\bigcup_{u \in \mathrm{path}(w_N)} \mathrm{atoms}(u)$, which CARS has excluded by Lemma A.1. A stream sample is a rejection of a strategy iff it is invalid and not in that strategy's excluded set at that time, and CARS's excluded set is larger. Valid samples are never skipped, so "before the $n$-th valid sample" means the same stream indices for all, and LM calls up to it are $n$ plus rejections. For the LM-call clock, CARS skips more, so its $k$-th call comes no earlier in the stream than $S$'s; every valid sample up to that point is one of CARS's $k$ calls, so CARS accepts at least as many of its first $k$ calls as $S$ does, and rejects no more. □

## B SHARPER CLOCKS AND FINER RESOLUTION

Theorem 4.1 uses the crudest bounds on the mass $p_\varepsilon$ left in the proposal: $p_\varepsilon \leq 1$ for CARS and $p_\varepsilon \geq P(\mathcal{L})$ for ARS. This appendix replaces them by bounds that follow $p_\varepsilon$ along a run, and lets the two masses be read at any resolution. Theorem 4.1 sorts every prefix $v$ by its *reach probability* $P(v)$, the chance that a draw from $P$ passes through $v$, against one threshold $t$, and every atom $(v, a)$ by its *mass* $P(va)$, the chance of the bad token $a$ right after $v$, against one threshold $s$. That is the coarsest rounding: every prefix counts as reached or not, every atom as light or heavy. Here each reach probability may be rounded down, and each atom mass rounded up, to any chosen set of values, from those two numbers all the way to the exact per-atom trie. The evaluation in Section 5 uses these sharper forms.

**Two clocks.** A *clock* is the unit in which a run's progress is counted: Section 3 counts valid samples, while Section 4 and this appendix count LM calls, $k$, accepted or not (Section C compares the two). What a budget of $k$ calls buys each algorithm is not $k$ draws from $P$. A call is a draw from $P$ with the excluded set removed and the rest renormalized by $p_\varepsilon(\mathcal{W})$, so it reaches a prefix $v$ with probability $P(v)/p_\varepsilon(\mathcal{W})$ and hits an atom of mass $m$ with probability $m/p_\varepsilon(\mathcal{W})$. How large these are depends on $p_\varepsilon$ along the run, and the two algorithms differ exactly there: CARS drives $p_\varepsilon$ down, ARS barely moves it. An *envelope* is a fixed sequence of numbers, one per call, that bounds $p_\varepsilon$ on *every* run: $c_i \geq p_\varepsilon(\mathcal{W}_i)$ after $i$ calls of CARS, or $\ell_i \leq p_\varepsilon(\mathcal{W}_i)$ after $i$ calls of ARS. Both below are computed from the trie alone, and they make the per-call probabilities above usable:

$p_\varepsilon(\mathcal{W}_i)$ depends on how the run went so far, but with the envelope in its place call $i$ reaches $v$ with probability at least $P(v)/c_i$ under CARS, and hits an atom of mass $m$ with probability at most $m/\ell_i$ under ARS, numbers known before the run.

- *The forced chain* (CARS). Let $F_0 = \varepsilon$ and let $F_{j+1}$ be the unique grammatical child of $F_j$, for as long as there is one. The chain is the opening the grammar leaves no choice about, such as a mandatory keyword, and CARS learns one link of it per call. Call $j$ starts at $\varepsilon$, whose bad tokens are excluded after call 1, is forced to $F_1$, whose bad tokens are excluded after call 2, and so on: it reaches $F_{j-1}$ and excludes its bad tokens whether or not it goes wrong there. So after $i$ calls, on every run, $p_\varepsilon(\mathcal{W}_i) \le c_i := 1 - \sum_{j<i} P(F_j)P(F_j\mathrm{bad}_{F_j} \mid F_j)$.
- *The atom count* (ARS). A call removes at most one atom, so after $i$ calls, on every run, $p_\varepsilon(\mathcal{W}_i) \ge \ell_i := \max\big(P(\mathcal{L}),\, 1 - M_i\big)$, where $M_i$ is the mass of the $i$ heaviest atoms. At best ARS removes the heaviest atoms first, one per call, and even then $p_\varepsilon$ can fall no faster than this.

Theorem A.2(a),(c) bound the probability that a prefix is still unvisited, or an atom still unhit, after $k$ calls by a $k$-th power; applying the one-call bound call by call, with the envelope in place of $p_\varepsilon$ at each call, turns the power into a product over the calls (Section C). Write

$$\varphi(x) = \prod_{i<k} \Big(1 - \frac{x}{c_i}\Big)_+, \qquad \psi(x) = \prod_{i<k} \Big(1 - \frac{x}{\ell_i}\Big), \qquad (x)_+ = \max(x, 0),$$

for the two *survival profiles*: after $k$ calls CARS has not reached a prefix of probability $x$ with probability at most $\varphi(x)$, and ARS has not hit an atom of mass $x \le s_0 := \min_{i<k} \ell_i$ with probability at least $\psi(x)$. Here $s_0$ is the lowest the ARS envelope goes within the budget, and heavier atoms fall outside the bound. (The factor $(\cdot)_+$ is for the nodes $F_j$ of the chain themselves, which are cleared outright.) Since $\varphi(x) \le e^{-xK_k^{\mathrm{CARS}}}$ with $K_k^{\mathrm{CARS}} = \sum_{i<k} 1/c_i \ge k$, and $\psi(x) \approx e^{-xK_k^{\mathrm{ARS}}}$ with $K_k^{\mathrm{ARS}} = \sum_{i<k} 1/\ell_i \le k/P(\mathcal{L})$, the $k$ calls of CARS are worth $K_k^{\mathrm{CARS}}$ draws from $P$ and the $k$ calls of ARS at most $K_k^{\mathrm{ARS}}$. These are the two *effective clocks*: $\varphi$ drops from 1 to 0 around $x \approx 1/K_k^{\mathrm{CARS}}$, the probability a prefix must carry for CARS to expect to reach it within the budget, and $\psi$ around $x \approx 1/K_k^{\mathrm{ARS}}$, the mass an atom must carry for ARS to expect to draw it. In the running example at $k = 10$ the root's bad tokens (0.30) are cleared by the first call, so $c_i = 0.70$ from then on and $K_k^{\mathrm{CARS}} = 13.9$ against $k = 10$, while the ten heaviest atoms weigh 0.13, so $K_k^{\mathrm{ARS}} = 11.2$ against $k/P(\mathcal{L}) = 40$ (Section C). The clocks move furthest from $k$ and $k/P(\mathcal{L})$ when the root is bad and $c_1 = 1 - P(\varepsilon\mathrm{bad}_\varepsilon \mid \varepsilon)$ is tiny; Section 5 reports them on the benchmarks.

**Any rounding.** Theorem 4.1 sorts every prefix by one threshold $t$ and every atom by one threshold $s$. The same argument allows any rounding: round every prefix's reach probability *down*, to some $\tau(v) \le P(v)$, and every atom's mass *up*, to some $\sigma(v, a) \ge P(va)$, dropping the atoms that cannot be rounded to a value below $s_0$.

**Theorem B.1** (Scope, at any resolution)**.** *For every budget $k$ and every such rounding $(\tau, \sigma)$,*

$$\mathrm{Speedup}(k) \;\ge\; \max\left(1,\; \frac{\overbrace{P(\mathcal{L}) \;+\; \sum_{(v,a)\ \text{kept}} P(va)\,\psi\big(\sigma(v,a)\big)}^{\le\, \mathbb{E}\, p_\varepsilon \text{ under ARS: mass still in its proposal}}}{\underbrace{1 \;-\; \sum_{v} P(v)P(v\mathrm{bad}_v \mid v)\,\Big(1 - \varphi\big(\tau(v)\big)\Big)}_{\ge\, \mathbb{E}\, p_\varepsilon \text{ under CARS: mass still in its proposal}}}\right).$$

Both sides of the fraction bound the same quantity for the two algorithms: the mass still in the proposal after $k$ calls, $p_\varepsilon(\mathcal{W}_k)$, which is the valid mass plus whatever invalid mass the algorithm has not yet removed; a call is accepted with probability $P(\mathcal{L})/p_\varepsilon(\mathcal{W}_k)$, so their ratio is the speedup. The numerator bounds it from below for ARS, the denominator from above for CARS. The proof is one sentence: a prefix is at least as reachable as the value it is rounded down to, and an atom at least as light as the value it is rounded up to, so the rounding only lowers the bound (`expected_pε_le_rounded_env`, `expected_pε_ge_rounded_env`, Section H); the max with 1 is Theorem A.4. Section C gives the per-node bounds and the envelopes.

**The spectrum.** Three roundings are worth naming. With no rounding at all, $\tau(v) = P(v)$ and $\sigma(v, a) = P(va)$, the theorem keeps one term per prefix and one per atom, the form of (3) in Section C: the tightest, but with as many terms as the trie has prefixes and atoms, nothing a reader can look at. With the edges of a geometric grid, $1, r^{-1}, r^{-2}, \ldots$, it becomes a pair of histograms. Cut the invalid mass into bins by reach probability and by atom mass, the $j$-th bin holding

$$R_j = \sum_{v:\ r^{-j-1} \le P(v) < r^{-j}} P(v)\, P(v\mathrm{bad}_v \mid v), \qquad \Lambda_j = \sum_{(v,a):\ s_0 r^{-j-1} \le P(va) < s_0 r^{-j}} P(va),$$

and weight each bin by the survival profile at its least reachable prefix, $\varphi(r^{-j-1})$, or at its heaviest atom, $\psi(s_0 r^{-j})$. With a single edge on each side, $t$ and $s$, the histograms collapse to two numbers, $R_t$, the invalid mass at prefixes the LM reaches with probability at least $t$, and $\Lambda_s$, the invalid mass in atoms lighter than $s$; the prefixes below $t$ are rounded to 0 and counted as never reached, and the atoms above $s$ are dropped.

| rounding | CARS side (subtracted in the denominator) | ARS side (added in the numerator) | reads as |
|---|---|---|---|
| none | $\sum_v P(v)P(v\mathrm{bad}_v \mid v)\,(1-\varphi(P(v)))$ | $\sum_{(v,a):\,P(va)\le s_0} P(va)\,\psi(P(va))$ | the trie |
| grid, ratio $r$ | $\sum_{j\ge 0}(1-\varphi(r^{-j-1}))\,R_j$ | $\sum_{j\ge 0}\psi(s_0 r^{-j})\,\Lambda_j$ | two histograms |
| one edge, $t$ and $s$ | $(1-\varphi(t))\,R_t$ | $\psi(s)\,\Lambda_s$ | two numbers |

Coarser is looser, and the loss is easy to see: a bin charges every prefix in it as if it sat at the bin's lower edge, so on the CARS side a grid of ratio $r$ stays within a factor $r$ of the per-atom form on every term; on the ARS side each bin keeps only the survival of its heaviest atom, and the top bin, weighted by $\psi(s_0) = 0$, is written off. A single edge charges everything above $t$ as if it were exactly at $t$ and writes off everything below. Section 5 measures the three against each other.

## C THE LM-CALL CLOCK

We call the unit in which a run's progress is counted a *clock*. Section 3 counts in the *valid-sample clock*, which ticks once per accepted sample and in which the answers are exact; Section 4 and this appendix count in the *LM-call clock*, which ticks once per call, accepted or not. The paper's iteration count is the number of LM calls, $k$. In this clock no closed form is available: the state after $k$ calls is that of the stream after a random number $N_k \ge k$ of samples, and the bounds below use only that $N_k \ge k$. Let $\mathrm{bad}_v^k \subseteq \mathrm{bad}_v$ be the bad tokens at $v$ still unpruned after $k$ LM calls. The identity

$$\mathbb{E}\big[p_\varepsilon(\mathcal{W}_k)\big] = P(\mathcal{L}) + \sum_v P(v)\,\mathbb{E}\big[P(\mathrm{bad}_v^k \mid v)\big] \tag{1}$$

(`expected_pε_eq_atoms`) holds for both algorithms, and in the valid-sample clock the same identity with Theorem A.2(b),(c) is exact:

$$\begin{aligned} \mathbb{E}_{\mathrm{CARS}}\big[p_\varepsilon\big] &= P(\mathcal{L}) + \sum_v P(v)P(v\mathrm{bad}_v \mid v)\,\rho_v^n, \\ \mathbb{E}_{\mathrm{ARS}}\big[p_\varepsilon\big] &= P(\mathcal{L}) + \sum_{(v,a)} P(va)\Big(\frac{P(\mathcal{L})}{P(\mathcal{L}) + P(va)}\Big)^n. \end{aligned} \tag{2}$$

In the LM-call clock Theorem A.2(a),(c) give, without envelopes,

$$\begin{aligned} \mathbb{E}_{\mathrm{CARS}}\big[P(\mathrm{bad}_v^k \mid v)\big] &\le P(v\mathrm{bad}_v \mid v)\,(1-P(v))^k, \\ \mathbb{E}_{\mathrm{ARS}}\big[P(\mathrm{bad}_v^k \mid v)\big] &\ge \sum_{a\in\mathrm{bad}_v} P(va \mid v)\Big(1 - \frac{P(va)}{P(\mathcal{L})}\Big)_+^k \end{aligned}$$

(`expected_pε_le_atoms_strong` applied with the sets $\mathrm{atoms}(v)$ as atoms, and `expected_pε_ge_atoms`). At state $\mathcal{W}$ a valid sample costs $p_\varepsilon(\mathcal{W})/P(\mathcal{L})$ LM calls on average; define $\mathrm{Speedup}(k)$ as the ratio of the expected costs at iteration $k$, the factor by which ARS needs more LM calls per valid sample than CARS after $k$ calls. By the identity it is the ratio of the

two expected masses, each a sum over prefixes:

$$\begin{aligned}\mathrm{Speedup}(k) \;&=\; \frac{P(\mathcal{L}) + \sum_v P(v)\,\mathbb{E}_{\mathrm{ARS}}\big[P(\mathrm{bad}_v^k \mid v)\big]}{P(\mathcal{L}) + \sum_v P(v)\,\mathbb{E}_{\mathrm{CARS}}\big[P(\mathrm{bad}_v^k \mid v)\big]} \\ &\geq\; \frac{P(\mathcal{L}) + \sum_v P(v)\cdot\sum_{a\in\mathrm{bad}_v} P(va \mid v)\big(1 - P(va)/P(\mathcal{L})\big)_+^k}{P(\mathcal{L}) + \sum_v P(v)\cdot P(v\mathrm{bad}_v \mid v)\,(1-P(v))^k}\,.\end{aligned} \tag{3}$$

The first line is exact; the second substitutes the two bounds above and is machine-checked term by term. Its reading: in the numerator a bad token $a$ at $v$ keeps its full weight $P(va \mid v)$ while it is light, $P(va) \lesssim P(\mathcal{L})/k$, and fades once ARS is likely to have drawn it; in the denominator the whole of $P(v\mathrm{bad}_v \mid v)$ disappears once $v$ is visited, $P(v) \gtrsim 1/k$. The speedup is at least 1 (Theorem A.4) and at most $1/P(\mathcal{L})$, and it is transient: ARS eventually collects the light tokens too. The valid-sample-clock version of the gap is Corollary C.1.

**Envelopes.** Both one-step facts behind the two bounds are exact: from a state $\mathcal{W}$ that has not touched $v$, the next call passes through $v$ with probability exactly $P(v)/p_\varepsilon(\mathcal{W})$, and it hits an unremoved atom of mass $m$ with probability exactly $m/p_\varepsilon(\mathcal{W})$. The bounds above use only $p_\varepsilon(\mathcal{W}) \leq 1$ and $p_\varepsilon(\mathcal{W}) \geq P(\mathcal{L})$. If $c_i \geq p_\varepsilon(\mathcal{W}_i)$ on every CARS run and $\ell_i \leq p_\varepsilon(\mathcal{W}_i)$ on every ARS run, conditioning on the state at each call gives

$$\begin{aligned}\mathbb{P}[v \text{ unreached after } k \text{ calls}] &\leq \prod_{i<k}\big(1 - P(v)/c_i\big)_+ && (\texttt{surv_le_prod_env}),\\ \mathbb{P}[A_{v,a} \text{ unhit after } k \text{ calls}] &\geq \prod_{i<k}\big(1 - P(va)/\ell_i\big) && (\texttt{prod_le_surv_env}),\end{aligned}$$

the second whenever $P(va) \leq \ell_i$ for every $i < k$; and (3) holds with these products in place of the powers, which is the form of Section B, evaluated in Section 5. In Lean an envelope is a predicate on (number of calls so far, excluded set) that holds at the start, is preserved by every draw of positive probability, and bounds $p_\varepsilon$; the two of Section B are instances. *The forced chain:* with $D_j = F_j\Sigma^*\$$ and $B_j = \mathrm{atoms}(F_j)$, every draw in $D_j$ prunes $B_j$ and $D_j \setminus B_j = D_{j+1}$, since $F_{j+1}$ is the only grammatical continuation of $F_j$; so a draw of positive probability, which avoids $B_0, \ldots, B_{i-1}$ once they are excluded, lies in $D_i$ and prunes $B_i$, and after $i$ calls $p_\varepsilon \leq 1 - P(B_0 \cup \cdots \cup B_{i-1}) = c_i$ (`chainEnv_step`, `chainEnv_bound`). *The atom count:* under ARS a draw prunes at most the one atom containing it, so after $i$ calls the excluded set lies in a union of at most $i$ atoms and $p_\varepsilon \geq 1 - M_i$; with $p_\varepsilon \geq P(\mathcal{L})$ this is $\ell_i$ (`arsEnv_step`, `arsEnv_bound`). Splitting the sums at a threshold gives the two bounds of Theorem 4.1, and with these products in place of the powers, Theorem B.1.

**Corollary C.1** (The gap)**.** *With $\Lambda_s$ the invalid mass in atoms of mass below $s$ (Section 4),*

$$\mathbb{E}\,\mathrm{Calls}_{\mathrm{ARS}}(n) - \mathbb{E}\,\mathrm{Calls}_{\mathrm{CARS}}(n) \;\geq\; \frac{n}{2\,P(\mathcal{L})}\,\Lambda_{P(\mathcal{L})/n} \;-\; \sum_v P(v\mathrm{bad}_v \mid v).$$

*Proof.* For $x = P(va)/P(\mathcal{L}) \leq 1/n$, Bernoulli's inequality gives $1-(1+x)^{-n} \geq nx/(1+nx) \geq nx/2$; sum over the light atoms, drop the others, and use Corollary 3.2 for CARS. □

Over $n$ valid samples ARS behaves like rejection sampling on the atoms lighter than $P(\mathcal{L})/n$—it pays for their mass at the RS rate $1/P(\mathcal{L})$—while CARS's total waste is independent of $n$.

**Corollary C.2** (ARS's waste splits in two)**.** *Let $\theta = P(\mathcal{L})/n$. Then*

$$\mathbb{E}\,\mathrm{Rej}_{\mathrm{ARS}}(n) \;\leq\; \big|\{(v,a) : P(va) \geq \theta\}\big| \;+\; \frac{n}{P(\mathcal{L})}\,\Lambda_\theta.$$

*Proof.* Split the sum of Theorem 3.1 at $\theta$. Each term is at most 1, which bounds the heavy part by their number; and for $m = P(va) < \theta$, Bernoulli's inequality gives $1-(1+m/P(\mathcal{L}))^{-n} \leq nm/P(\mathcal{L})$, which summed over the light atoms is the second term. □

The two terms are the two ways ARS loses: one rejected call for each atom heavy enough that it has been drawn by now, and the RS rate on all the mass that has not. Corollary C.1 bounds the second term only, which is why it is loose where the first dominates. CARS has no second term—its total is $\sum_v P(v\mathrm{bad}_v \mid v)$ whatever $n$ is—and that asymmetry is the speedup.

## D THE RUNNING EXAMPLE, CONTINUED

**The LM-call clock.** Simulating both algorithms on the running example and evaluating the bounds of Section C (2000 runs):

| | $\mathbb{E}p_\varepsilon$, CARS | | $\mathbb{E}p_\varepsilon$, ARS | | | LM calls per valid sample | | |
|---|---|---|---|---|---|---|---|---|
| $k$ | simulated | bound | simulated | bound | Speedup | CARS | ARS | RS |
| 1 | 0.56 | 0.56 | 0.99 | 0.95 | 1.8 | 2.2 | 4.0 | 4.0 |
| 2 | 0.42 | 0.47 | 0.98 | 0.92 | 2.3 | 1.7 | 3.9 | 4.0 |
| 5 | 0.29 | 0.36 | 0.95 | 0.88 | 3.3 | 1.2 | 3.8 | 4.0 |
| 10 | 0.26 | 0.30 | 0.92 | 0.85 | 3.5 | 1.1 | 3.7 | 4.0 |
| 20 | 0.25 | 0.27 | 0.88 | 0.80 | 3.5 | 1.0 | 3.5 | 4.0 |

The first CARS draw removes the bad tokens at the root (0.30) and, with probability 0.50, those at `0` (0.20), and so on down its path: $\mathbb{E}p_\varepsilon$ drops from 1 to 0.56 in one call. After ten calls CARS is at $p_\varepsilon \approx P(\mathcal{L})$ and accepts almost every draw. ARS has in the meantime collected the heavy atom `0++` and a few of the 900 light tokens, at most one per call; its remaining mass barely moves, because each light token is hit with probability about $m/p_\varepsilon \approx 0.003$ per call. The speedup peaks at 3.5, against the ceiling $1/P(\mathcal{L}) = 4$. The bounds are within 0.07 of the simulation on the CARS side; the ARS bound is looser because it credits ARS with the best-case renormalization $p_\varepsilon \geq P(\mathcal{L})$.

**The spread (Theorem A.3).** By `var_check.py` (6000 runs): CARS's total waste is a sum of nine independent Bernoulli variables, one per node (a Poisson-binomial distribution), with mean 2.9 and variance 1.68 (simulated: 1.68 at $n = 60$); it is at most 4 with probability 0.89 and the simulated and predicted probability mass functions agree to $\pm 0.004$. ARS's rejections before the fifth valid sample have mean 13.5 and variance 43.9 (simulated: 13.5 and 44.7), against 12.3 if the atoms were hit independently.

## E THE BENCHMARKS OF THE CARS PAPER BY KIND

Sorting the benchmarks and observations reported by Parys et al. (2026) by the kinds of Section 4:

| Benchmark | Structure | Observed |
|---|---|---|
| SyGuS (BV4, SLIA) | Format and tight per-position. 1–20 of 128K tokens admissible at every position; at the root the LM puts 27.7% and 8.0% on two invalid tokens and the rest of the invalid root mass on a tail of 128K; deeper, 2–8% per position. Theorem A.2(c): ARS needs $\geq$ 128,255 rejections for the root alone; CARS clears it on draw 1. | RS produces nothing within $10^4$ calls. CARS uses $16\times$ (BV4) and $4.5\times$ (SLIA) fewer calls than ARS. |
| SMILES | Tight per-position, with an aligned LM: same $\lvert\mathrm{bad}_v\rvert$, but few-shot exemplars make $P(v\mathrm{bad}_v \mid v)$ small and concentrated on a few tokens that ARS finds too. | $4.3\times$ fewer samples than RS, $1.3\times$ fewer than ARS. |
| Spider (text-to-SQL) | Loose grammar: aligned LM, $P(\mathcal{L}) \approx 0.5$, a fresh trie per query, stopped at its first valid sample. Ceiling $1/P(\mathcal{L}) \approx 2$. | $1.85\times$ fewer than RS, $1.28\times$ fewer than ARS. |
| Fuzzing seeds (JSON, SQL, XML) | Format and global mixed: syntax is shallow and tight; the counting requirements are violated deep, at many distinct prefixes. | CARS averages 2.15 calls per valid XML sample. |
| PDDL | Severe misalignment on a spine, hence format (Llama-8B: wrong output format, $P(\mathcal{L})$ tiny but mistakes shallow), or loose grammar (Qwen, aligned). | Llama-8B: RS and ARS produce nothing within the cap, 61% of CARS's calls are valid. Qwen-7B: $1.2\times$ fewer calls than ARS. |
| SQL with mandated `set ::timeout` | Severe misalignment at many frontier nodes, hence global: the directive is missing at many diverse prefixes. | RS 0%, CARS $< 0.1\%$ acceptance. |

# F THE EVALUATION IN FULL

This appendix gives what Section 5 leaves out: how each quantity is measured, the per-task detail behind Tables 1 and 4, the comparison across checkpoints (the model axis, Table 9), and how the predictions evolve during a run. Section G has the experimental details.

**What we ran.** One task per benchmark family, with the grammar and prompt of the CARS repository: SMILES (acrylates) (Weininger, 1988; Guo et al., 2022), SyGuS SLIA and BV4 (Alur et al., 2019), and the JSON, XML and SQL fuzzing grammars of Parys et al. (2026), the last three also with the repository's second prompt, which states the grammar in Lark before the examples ("grammar in prompt"). The LM is Llama-3.1-8B-Instruct in `bfloat16` at temperature 1 with no truncation (Section G). Each strategy runs from the same prompt until 100 valid samples or its budget; CARS runs ten times and ARS five, and RS five times on SMILES, SLIA and BV4, at the same 8,000-call budget as ARS. The model axis (Table 9) repeats the CARS and ARS cells on the six grammars against six more checkpoints—the base Llama-3.1-8B, Qwen-2.5 7B and 14B, Qwen-2.5-Coder-7B, Qwen-2.5-Math-7B and phi-4—and the grammar-in-prompt variants against Qwen-2.5-7B, with a heavy-prefix enumeration for every pair. CARS gets more runs because one of them supplies the trie, leaving nine independent of the predictions, and ARS fewer because its runs are far longer (3,341 calls on BV4 against 177 for CARS); on the fuzzing grammars $P(\mathcal{L})$ leaves RS no chance, so it runs only on SMILES, SLIA and BV4.

**How the quantities are measured.** Theorem 3.1 needs three things per prefix—$P(v)$, $P(v\mathrm{bad}_v \mid v)$ and the atom masses $P(va)$—and one global constant, $P(\mathcal{L})$. The first three are not estimated at all; they are *read off*, exactly up to `float32` rounding. At every prefix it visits, CARS already holds the LM's next-token distribution and the oracle's mask of the invalid tokens there, which give $P(v\mathrm{bad}_v \mid v)$, $\lvert\mathrm{bad}_v\rvert$ and every $P(va)$ directly, while $P(v)$ is the product of the conditionals along the path. No sampling noise enters any of them; a run chooses only *which* prefixes appear, and CARS visits them in proportion to how often the LM goes there, the weighting the theorems need (Section G).

What must be estimated is global. A call at state $\mathcal{W}$ is accepted with probability $P(\mathcal{L})/p_\varepsilon(\mathcal{W})$ and the sampler knows $p_\varepsilon(\mathcal{W})$, so $\hat{P}(\mathcal{L}) = \#\text{accepted}/\sum_{\text{calls}} 1/p_\varepsilon(\mathcal{W})$ is consistent for every strategy

and every state it passes through ( a ratio of two sums, consistent but not unbiased; Section G). We use the seed-0 CARS run alone, so that no run being predicted also feeds the prediction: 100 accepted samples on every task, a standard error of about 10%. The $\rho_v$ also need $P(v \mid \mathcal{L})$, which we take as the fraction of that run's accepted samples through $v$—legitimate because CARS draws exactly from $P^{\mathcal{L}}$, so its accepted samples are i.i.d. draws from it. Finally, a trie says nothing about prefixes its run never reached. That mass we do not estimate but bracket. The lower end of every range is Theorem 3.1 summed over the atoms the trie holds; every term it leaves out is nonnegative, so it can only undercount. The upper end adds the unseen invalid mass $m$ charged at the RS rate, $n\,m/P(\mathcal{L})$ calls for $n$ valid samples, which bounds the terms left out: an unseen atom of mass $m$ costs ARS $1-(P(\mathcal{L})/(P(\mathcal{L})+m))^n \le nm/P(\mathcal{L})$, and an unseen prefix costs CARS $P(v\mathrm{bad}_v \mid v)(1-\rho_v^n) \le n\,P(v)P(v\mathrm{bad}_v \mid v)/P(\mathcal{L})$, since $1-\rho^n \le n(1-\rho)$ and $P(v) \ge P(\mathcal{L} \cap v\Sigma^*\$)$. An adaptive sampler never pays more for mass it has not learned about than rejection sampling does. The unseen mass itself is a difference of two numbers close to 1,

$$m_{\text{unseen}} = 1 - \hat{P}(\mathcal{L}) - \sum_{v \text{ visited}} P(v)\,P(v\mathrm{bad}_v \mid v),$$

and the trie's masses are only as precise as the arithmetic that produces them: each $P(v\mathrm{bad}_v \mid v)$ is a `float32` sum of up to $|\Sigma| = 128{,}256$ probabilities from a `float32` log-softmax, with relative error at most $(\log_2|\Sigma|+2)\,2^{-24}$, so the mass seen is known only to within $\delta = (\log_2|\Sigma|+2)\,2^{-24}\sum_{v \text{ visited}} P(v)$, about $10^{-6}$ on our tries. The upper end therefore charges $\max(m_{\text{unseen}},0)+\delta$ at the RS rate. Where $\hat{P}(\mathcal{L})$ is far above $\delta$ this changes little: $\delta/\hat{P}(\mathcal{L})$ is below 0.05 calls per valid sample on SMILES, SLIA and BV4, and 1.7 on JSON. Where $\hat{P}(\mathcal{L})$ is below $\delta$, as on SQL and the grammar-in-prompt tasks, $\delta/\hat{P}(\mathcal{L})$ dominates, and the range says that the trie cannot resolve the cost. Predictions use the seed-0 CARS trie and measured costs are means over all runs, ten of CARS and five of ARS; the trie supplies only the node masses, and the quantity it predicts, the number of calls, is measured on runs nearly all independent of it: nine of the ten CARS runs and all five ARS runs.

**Where the prediction is sharp.** On SyGuS almost nothing is left unseen: the measured CARS cost lies inside the predicted range, and the ARS cost within 12% of it, the error of $\hat{P}(\mathcal{L})$. The LM puts 0.979 (SLIA) and 1.000 (BV4) of its root mass on invalid tokens, so $\rho_\varepsilon = 0$ and CARS clears the root on its first call. That is where the two lifetime bills of Corollary 3.2 are furthest apart: 128,255 bad tokens at the root, one ARS rejection each, against $P(\varepsilon\mathrm{bad}_\varepsilon \mid \varepsilon) \le 1$ for CARS. Clearing the root alone leaves an acceptance rate of only 0.042; the rest of the CARS cost is spread over the deeper prefixes: the lifetime total $\sum_v P(v\mathrm{bad}_v \mid v)$ is 19.5 (SLIA) and 84.0 (BV4), of which 100 valid samples pay 11–19 and 39–91 (Table 8). On JSON, XML and SMILES much of the invalid mass lies beyond the run—at the end of the seed-0 runs $p_\varepsilon$ is still 4 (XML) and 7 (JSON) times $\hat{P}(\mathcal{L})$—so the prediction is a range, and the measured cost sits in its upper part: on JSON and XML a global constraint on top of a format root, on SMILES the same deep, spread-out mass under a clean root.

**The four hardest tasks.** SQL and the grammar-in-prompt variants have the smallest valid mass: $\hat{P}(\mathcal{L}) = 2.6 \times 10^{-10}$ on SQL, and stating the grammar in the prompt *lowers* it on every fuzzing grammar, from $6.5 \times 10^{-7}$ to $5.8 \times 10^{-9}$ on JSON, from $1.2 \times 10^{-5}$ to $2.6 \times 10^{-8}$ on XML and from $2.6 \times 10^{-10}$ to $4.1 \times 10^{-12}$ on SQL. ARS collects at most six valid samples on any of them. CARS completes all four, and more cheaply with the grammar in the prompt than without it despite the smaller $P(\mathcal{L})$: 397 calls against 700 on JSON, 215 against 416 on XML and 1,249 against 2,901 on SQL. On SQL both algorithms had a budget of 8,000 calls, after 2,000 had left CARS short of 100 valid samples; all ten CARS runs reached 100, in 2,452–3,524 calls, and ARS found none. At these values of $\hat{P}(\mathcal{L})$ the trie cannot resolve the unseen mass, so the predictions are lower bounds (the ranges above).

**Theorem 4.1 at three resolutions.** Where the root is bad, the effective clocks of Section B are far from $k$ and $k/P(\mathcal{L})$: on SLIA, BV4 and JSON, $c_1 = 1 - P(\varepsilon\mathrm{bad}_\varepsilon \mid \varepsilon)$ is 0.02, $3 \times 10^{-4}$ and $5 \times 10^{-6}$, so at $k = 100$ $K_k^{\mathrm{CARS}}$ is 7,000, $3 \times 10^5$ and $2 \times 10^7$, while $K_k^{\mathrm{ARS}}$ is 5,200, 19,000 and 67,000 against $k/P(\mathcal{L}) = 1.2 \times 10^5$, $3.7 \times 10^6$ and $1.5 \times 10^8$. The measured speedup is the ratio of the ARS and CARS curves of Figure 3 at $k = 100$, not the ratio of total calls in Table 1, which is

| | | RS | | ARS | | CARS | |
|---|---|---|---|---|---|---|---|
| Benchmark | $\hat{P}(\mathcal{L})$ | measured | predicted | measured | predicted | measured | predicted |
| SMILES (acrylates) | 0.049 | 2,109 | 2,055 | 776 | 560–803 | 489 | 271–515 |
| SyGuS SLIA | $8.5 \times 10^{-4}$ | 6 / 8,000* | $1.2 \times 10^{5}$ | 991 | 876–883 | 115 | 111–119 |
| SyGuS BV4 | $2.7 \times 10^{-5}$ | 0 / 8,000* | $3.7 \times 10^{6}$ | 3,341 | 3,610–3,666 | 177 | 139–195 |
| JSON | $6.5 \times 10^{-7}$ | – | $1.5 \times 10^{8}$ | 29 / 8,000* | 15,420–16,174 | 700 | 105–858 |
| XML | $1.2 \times 10^{-5}$ | – | $8.1 \times 10^{6}$ | 2,827 | 2,291–2,600 | 416 | 119–429 |
| SQL | $2.6 \times 10^{-10}$ | – | $3.8 \times 10^{11}$ | 0 / 8,000* | $1.3 \times 10^{5}$–$5.7 \times 10^{5}$ | 2,901 | 236–$4.5 \times 10^{5}$ |
| JSON (grammar in prompt) | $5.8 \times 10^{-9}$ | – | $1.7 \times 10^{10}$ | 4 / 8,000* | 68,010–87,653 | 397 | 104–19,748 |
| XML (grammar in prompt) | $2.6 \times 10^{-8}$ | – | $3.8 \times 10^{9}$ | 6 / 8,000* | 35,367–39,670 | 215 | 104–4,407 |
| SQL (grammar in prompt) | $4.1 \times 10^{-12}$ | – | $2.4 \times 10^{13}$ | 0 / 8,000* | $1.3 \times 10^{5}$–$2.8 \times 10^{7}$ | 1,249 | 123–$2.8 \times 10^{7}$ |

Table 2: All nine tasks (Table 1 shows four). LM calls for 100 valid samples, measured beside predicted for each strategy. Measured costs are means over ten CARS runs and five each of ARS and RS; predictions are Theorem 3.1 from the trie of the seed-0 CARS run, and for RS $100/\hat{P}(\mathcal{L})$, which needs no trie. The lower end of a range is Theorem 3.1 on the atoms the trie saw; the upper end adds the unseen invalid mass at the rejection-sampling rate, and that mass is known only to within the `float32` precision of the trie's masses, about $10^{-6}$. *: budget exhausted before 100 valid samples; valid / calls.

| | CARS trie after 100 valid samples | | | invalid mass | mass at $k = 100$ | |
|---|---|---|---|---|---|---|
| Benchmark | prefixes | $\sum_v P(v\text{bad}_v \mid v)$ | atoms $\geq P(\mathcal{L})/100$ | seen / total | heavy nodes | light bad nodes |
| SMILES (acrylates) | 1,496 | 382.5 | 236 | 0.833 / 0.951 | 0.520 | 0.146 |
| SyGuS SLIA | 125 | 19.5 | 298 | 0.999 / 0.999 | 0.998 | 0.005 |
| SyGuS BV4 | 390 | 84.0 | 1,314 | 1.000 / 1.000 | 1.000 | 0.001 |
| JSON | 38,939 | 570.6 | 7,094 | 1.000 / 1.000 | 1.000 | 0.000 |
| XML | 34,089 | 314.0 | 845 | 1.000 / 1.000 | 1.000 | 0.000 |
| SQL | 130,670 | 2798.2 | 124,798 | 1.000 / 1.000 | 1.000 | 0.000 |
| JSON (grammar in prompt) | 23,050 | 359.9 | 52,555 | 1.000 / 1.000 | 1.000 | 0.000 |
| XML (grammar in prompt) | 22,235 | 115.8 | 21,626 | 1.000 / 1.000 | 1.000 | 0.000 |
| SQL (grammar in prompt) | 71,368 | 1182.3 | 127,725 | 1.000 / 1.000 | 1.000 | 0.000 |

Table 3: The seed-0 CARS trie: prefixes visited, the asymptotic CARS waste (Corollary 3.2), atoms of mass $\geq P(\mathcal{L})/100$, invalid mass seen against $1 - \hat{P}(\mathcal{L})$, and the two masses of Section 4 at $k = 100$.

cumulative and counted in the other clock. The rows below are Theorem B.1 at the three roundings of Section B, each optimized over its thresholds on the seed-0 trie:

| $k = 100$ | SMILES | SLIA | XML | BV4 | JSON |
|---|---|---|---|---|---|
| measured $\text{Speedup}(100)$ | 1.8 | 15 | 19 | 98 | 279 |
| Theorem B.1, no rounding | 1 | 12 | 17 | 55 | 247 |
| Theorem B.1, grid $r = 2$ | 1 | 11 | 14 | 43 | 217 |
| Theorem B.1, one edge | 1 | 4.6 | 8.7 | 12 | 138 |
| ceiling $1/P(\mathcal{L})$ | 21 | 1,170 | 81,000 | 37,000 | $1.5 \times 10^{6}$ |

Each coarsening costs a factor: without rounding the bound is within 1.1–1.8 of the measurement wherever the root is bad, the grid of ratio 2 loses another 1.1–1.3, and the single edge, which writes off every prefix below $t$ and every atom above $s$, is 2–8 below. SMILES returns 1 at every resolution: its root is clean, so $c_i \equiv 1$, and the atom count is a poor envelope for ARS there, charging it with the removal of the 100 heaviest atoms (0.58 of the mass) where after 100 calls it has removed 0.42.

**Where each benchmark's invalid mass sits.** Read in units of $P(\mathcal{L})$ (Table 4), $\Lambda_{s^*}$ orders the tasks exactly as the measured speedup does, each within a factor of about five of it; in absolute terms it would mislead, since on BV4 $\Lambda_{s^*} = 0.003$, which is 103 wasted calls per valid sample. Tables 6 and 7 show where the mass sits. SyGuS puts nearly all of it at the root and the rest on a short `define-fun` spine, in atoms that are heavy for ARS only because $P(\mathcal{L})$ is so small. SMILES is the aligned case: nothing at the root, the bulk in atoms between $P(\mathcal{L})/100$ and $P(\mathcal{L})$ that ARS collects after a few valid samples, hence the smallest $\Lambda_{s^*}$, the largest $U_{t^*}$ and a speedup of 1.8. The fuzzing grammars combine a root the LM never gets past on its own (on JSON followed by a second

| | Speedup(100) | | | | |
|---|---|---|---|---|---|
| Benchmark | measured | bound | $\Lambda_{s^*}/\hat{P}(\mathcal{L})$ | $U_{t^*}/\hat{P}(\mathcal{L})$ | ceiling $1/\hat{P}(\mathcal{L})$ |
| SMILES (acrylates) | 1.8 | 1 | 6.8 | 10.3 | 21 |
| SyGuS SLIA | 15 | 12 | 12 | 1.0 | 1,170 |
| SyGuS BV4 | 98 | 55 | 103 | 5.1 | 37,000 |
| JSON | 279 | 247 | 1,380 | 5.9 | $1.5 \times 10^6$ |
| XML | 19 | 17 | 60 | 3.4 | 81,000 |

Table 4: Theorem 4.1 against the measured speedup after $k = 100$ calls on the seed-0 runs. The bound is the per-atom form of Section B. The two masses are read at the thresholds $(t^*, s^*)$ that maximize the single-threshold bound with the effective clocks of Section B; they play the role of $U^{(k)}$ and $\Lambda^{(k)}$ of Section 4, whose thresholds $1/k$ and $P(\mathcal{L})/k$ ignore that CARS's calls are worth more than $k$ draws and ARS's fewer than $k/P(\mathcal{L})$.

spine node, {"name", with $P(v\mathrm{bad}_v \mid v) = 1$) with mass that is deep and light, at prefixes far below $1/K_k^{\mathrm{CARS}}$; that mass is $U_{t^*}$, and neither algorithm touches it.

**The model axis.** Table 9 repeats the comparison on seven checkpoints. Its last four columns come from enumeration alone, with no sampling, and the speedup bound computed from them tracks the measured speedup over three orders of magnitude; the pairs where CARS gains little are those the enumeration marks as having little heavy invalid mass. The axis also separates the two things a checkpoint can change. Qwen-7B and Qwen-14B share a vocabulary token for token, so every $\mathrm{bad}_v$ is the same set and only the masses move; between an instruct checkpoint and its base, the root alignment moves, and $P(\mathcal{L})$ with it.

### F.1 How the predictions evolve during a run

A run contributes two things to Theorem 3.1: the prefixes it has visited, whose quantities are exact once visited, and an estimate of $P(\mathcal{L})$. For each $n$ we restrict the final seed-0 trie to the prefixes the calls up to the $n$-th valid sample passed through—exactly the trie at that moment, since node quantities do not depend on when a node was visited—re-estimate $P(\mathcal{L})$ from those calls alone, and predict the cost of 100 valid samples (Figure 2, which does this after every valid sample).

**The trie is ready early.** Where the root is bad, the first call clears it and the next few walk the spine, so the prefixes that carry the invalid mass are visited within the first dozen calls; a longer run adds prefixes that carry little. On SLIA the run holds 43 prefixes after 16 calls and 125 after 115, and the unseen mass falls from 0.3 to 0.07 calls per valid sample. The CARS prediction, a sum over those prefixes, settles with them: on SLIA 106–136 after 16 calls narrows to 111–119 around the measured 115, and on BV4 121–281 narrows to 139–195 around 177. The bound on $\mathrm{Speedup}(100)$ is within 6% of its final value from ten valid samples on: 11.9 against 12.3 on SLIA, 52.5 against 55.3 on BV4, 246 against 247 on JSON, and 1 on SMILES.

**The constant is ready late.** ARS's cost counts the atoms heavier than about $P(\mathcal{L})/n$ and pays for the rest at the RS rate, so it moves with the estimate of $P(\mathcal{L})$, whose relative error is about $1/\sqrt{\text{accepted samples}}$. On BV4 the first 37 calls put $P(\mathcal{L})$ at $2.1 \times 10^{-5}$, against $2.7 \times 10^{-5}$ at the end of the run, and the ARS prediction at 4,265–4,425 against 3,341 measured; after 100 valid samples the prediction is 3,610–3,666, still 8% high, the error of an estimate from 100 accepted samples.

**Deep mass keeps the range open.** On SMILES and JSON the invalid mass is spread over prefixes the LM rarely reaches, so the trie keeps growing (on JSON, 3,151 prefixes after 50 calls and 38,939 after 661) and the range narrows slowly; on SMILES its width falls from 4.1 to 2.4 calls per valid sample, and the measured CARS cost is inside it from the second valid sample on. On JSON the run's own $\hat{P}(\mathcal{L})$ drifts down as rare valid completions turn up (0.88, 0.76, 0.69 and $0.65 \times 10^{-6}$ after 10, 25, 50 and 100 samples, settling on the $0.65 \times 10^{-6}$ of the whole run), which moves the whole range, and the measured cost enters it at the twelfth valid sample.

**When to stop.** Both remaining errors are known to the sampler: the width of the range is the unseen mass in units of $\hat{P}(\mathcal{L})$, and the error of $\hat{P}(\mathcal{L})$ is $1/\sqrt{\text{accepted samples}}$. A practitioner can stop the prediction run once both are small enough for the decision at hand; the scope question needs neither, since ten valid samples settle the bound.

## G EXPERIMENTAL DETAILS

**Models and software.** The nine-task tables use one checkpoint and the small-model validation below a second; the model axis of Table 9 adds six more. All are identified by their Hugging Face revision so the runs can be repeated against the same weights.

| Role | Model | Hugging Face id | Revision |
|---|---|---|---|
| all benchmarks | Llama-3.1-8B-Instruct | `meta-llama/Llama-3.1-8B-Instruct` | `0e9e39f2` |
| model axis | Llama-3.1-8B (base) | `meta-llama/Llama-3.1-8B` | `d04e592b` |
| model axis | Qwen-2.5-7B-Instruct | `Qwen/Qwen2.5-7B-Instruct` | `a09a3545` |
| model axis | Qwen-2.5-14B-Instruct | `Qwen/Qwen2.5-14B-Instruct` | `cf98f3b3` |
| model axis | Qwen-2.5-Coder-7B-Instruct | `Qwen/Qwen2.5-Coder-7B-Instruct` | `c03e6d35` |
| model axis | Qwen-2.5-Math-7B-Instruct | `Qwen/Qwen2.5-Math-7B-Instruct` | `ef9926d7` |
| model axis | phi-4 | `microsoft/phi-4` | `2db69c1c` |
| pipeline validation | CodeGen-350M-multi (fine-tuned) | `hsultanbey/codegen350multi_finetuned` | `main` |

Llama runs in `bfloat16` on a single GPU, and the validation model in `float32`, on CPU for the first trials and on a single GPU for the thirty-trial rerun. No number in this paper is a wall-clock measurement—every cost is counted in LM calls—so the hardware has only to hold the weights and one forward pass on the device and the trie in host memory. The harness runs on `transformers` 4.57.6 and `torch` 2.5.1 (CUDA 12.1) with the CARS implementation at commit `f46607f9` of its repository. Eleven of the fifteen SQL runs at the 8,000-call budget, the ten CARS runs and ARS seed 0, ran on a different machine, with `torch` 2.6.0 and `transformers` 5.17.0; the driver is byte-identical, the sampler is the one used for every other run, and it never calls `generate`, drawing at temperature 1 from the model's next-token distribution, so no library defaults enter.

**Sampling.** The CARS code builds a generation configuration without temperature or truncation parameters, which `transformers` 4.57 completes from the model's defaults (for Llama-3.1, temperature 0.6, top-$p$ 0.9, top-$k$ 50). We set temperature 1, top-$p$ 1 and top-$k$ 0 on the model and assert the effective configuration at start-up; setting top-$k$ to *none*, the natural reading of "off", leaves it unset and the library fills in 50, which is how our first runs failed the assertion. For every LM call we log the tokens drawn, the acceptance and $p_\varepsilon(\mathcal{W})$ before the call. At the end of a run we dump the trie: for every visited grammatical prefix, $P(v)$, $P(v\text{bad}_v \mid v)$, $|\text{bad}_v|$, the 64 heaviest bad tokens and a log-binned histogram of the rest, so that no atom is dropped. Budgets are 1,000 LM calls and 256 new tokens on SMILES and 2,000 and 512 elsewhere for CARS, raised to 8,000 on SQL, and 8,000 calls for ARS throughout; the JSON CARS trie needs about 50 GB of host memory.

**The estimator $\hat{P}(\mathcal{L})$.** With $A_k$ the acceptance indicator of call $k$ and $w_k = 1/p_\varepsilon(\mathcal{W}_k)$, $\mathbb{E}[A_k \mid \mathcal{W}_k] = P(\mathcal{L})\, w_k$, so $A_k - P(\mathcal{L})w_k$ has conditional mean zero given the run so far (a martingale difference) and $\mathbb{E}[\sum_k A_k] = P(\mathcal{L})\,\mathbb{E}[\sum_k w_k]$. The identity does not transfer to the ratio: $\hat{P}(\mathcal{L}) = \sum_k A_k / \sum_k w_k$ is consistent, with a bias of order $1/K$. Estimated by leaving out one run at a time (the jackknife), the bias is $-0.04\%$ (SMILES), $+0.02\%$ (SLIA) and $-0.39\%$ (BV4), an order of magnitude below the standard error, so we do not correct for it. The per-strategy estimates agree within their standard errors.

**Checks on the runs.** Three checks come for free. The seed-0 SMILES run rejected 379 calls and its trie sums to $\sum_v P(v\text{bad}_v \mid v) = 382.5$, as Theorem A.3(a) says it should. ARS's cost is proportional to $1/\hat{P}(\mathcal{L})$, whose standard error from the 100 accepted samples of one run is about 10%, so its 8–12% error on SyGuS is about one standard error. And the spread over runs (Table 8) follows the unseen mass: where nothing is unseen (SLIA) the measured standard deviation of $\text{Rej}(100)$ is nearest the prediction—2.8 against 1.8 (1.6×) for CARS, and 27, 3% of the mean, for ARS—and the excess grows as the unseen mass grows: 1.7× (BV4), 4× (SMILES), 9× (XML) and 26×

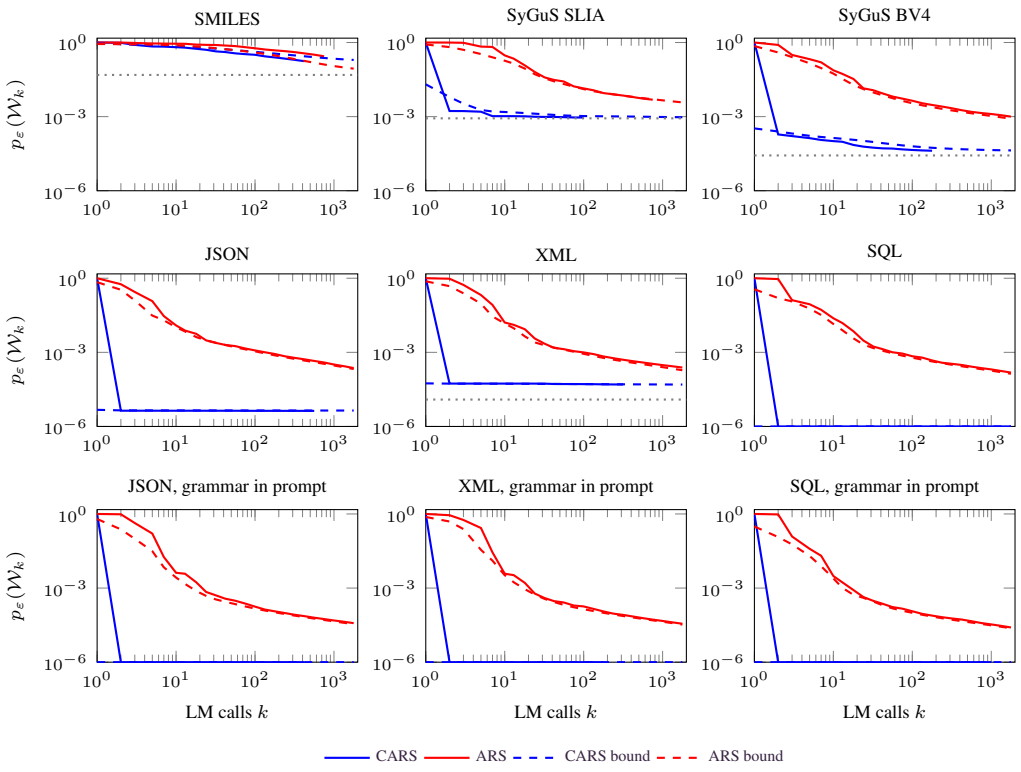



Figure 3: $p_\varepsilon(\mathcal{W}_k)$ against LM calls $k$ (log–log): measured for CARS and ARS, the CARS upper bound and ARS lower bound of Section C (with the envelopes $c_i$, $\ell_i$) evaluated on the trie, and $\hat{P}(\mathcal{L})$ (dotted).

(JSON), because the unseen part is paid at the RS rate, whose variance the trie cannot see. There the CARS prediction is the variance of Theorem A.3(a), a sum of independent Bernoulli variables over the visited prefixes, and the ARS prediction the variance of Theorem A.3(b); five runs estimate a standard deviation only to within a factor of about two.

**The acceptance curves.** Figure 3 shows $p_\varepsilon(\mathcal{W}_k)$ against LM calls on the seed-0 runs, with the bounds of Section C. Where the root is invalid the CARS curve falls by three to five orders of magnitude at the first call and creeps down from there, while ARS falls by about a factor of ten for every tenfold increase in calls, one atom at a time; the ARS bound is loose but never crossed. The speedup in this clock peaks within seven calls at 630 (SLIA), 4,200 (BV4), 18,000 (XML) and 130,000 (JSON), against ceilings $1/P(\mathcal{L})$ of 1,200, 37,000, 81,000 and $1.5 \times 10^6$; on SMILES, where the root is clean, it climbs slowly to 1.8 at $k = 100$ against a ceiling of 21. Tables 5 to 8 give the rejections before the $n$-th valid sample, where the invalid mass sits, the heaviest prefixes, and the spread.

**Validation on a small model.** Before the benchmark runs we validated the harness end to end with the CPU-sized model of the CARS repository (CodeGen-350M) on a task in the spirit of the running example: $E ::= d \mid d + E$ over the digits, followed by a newline. The model has $P(\mathcal{L}) \approx 0.60$ and makes mistakes both at the root (`np`, `x`, `-`) and deeper (`1 -`, `1 + 1.`); on the benchmarks it has essentially no valid mass, which is why those need the 7–14B models. Predictions use only run data, as on the benchmarks. Over thirty independent trials to the 100th valid sample:

| | $n = 1$ | $n = 10$ | $n = 20$ | $n = 50$ | $n = 100$ |
|---|---|---|---|---|---|
| *SMILES (acrylates)* | | | | | |
| CARS measured | 11 | 78 | 146 | 253 | 389 |
| CARS predicted | 11–13 | 59–83 | 87–136 | 134–255 | 171–415 |
| ARS measured | 13 | 119 | 229 | 420 | 676 |
| ARS predicted | 14–17 | 100–124 | 165–214 | 304–425 | 460–703 |
| RS measured | 27 | 252 | 468 | 1,021 | 2,009 |
| RS predicted | 20 | 196 | 391 | 978 | 1,955 |
| *SyGuS SLIA* | | | | | |
| CARS measured | 2 | 4 | 6 | 10 | 15 |
| CARS predicted | 3 | 5–6 | 6–8 | 9–12 | 11–19 |
| ARS measured | 68 | 207 | 350 | 591 | 891 |
| ARS predicted | 53 | 187–187 | 280–281 | 495–498 | 776–783 |
| RS measured | 1,181 | – | – | – | – |
| RS predicted | 1,173 | 11,726 | 23,453 | 58,632 | 117,264 |
| *SyGuS BV4* | | | | | |
| CARS measured | 5 | 20 | 33 | 51 | 77 |
| CARS predicted | 6–6 | 18–23 | 23–34 | 32–58 | 39–91 |
| ARS measured | 221 | 784 | 1,162 | 2,103 | 3,241 |
| ARS predicted | 163–164 | 735–740 | 1,172–1,182 | 2,190–2,216 | 3,510–3,562 |
| RS measured | – | – | – | – | – |
| RS predicted | 37,284 | 372,845 | 745,690 | 1,864,224 | 3,728,448 |
| *JSON* | | | | | |
| CARS measured | 5 | 63 | 134 | 303 | 600 |
| CARS predicted | 1–7 | 2–60 | 2–118 | 4–293 | 5+ |
| ARS measured | 625 | 3,502 | 5,426 | – | – |
| ARS predicted | 782–788 | 3,734–3,792 | 5,783–5,899 | 10,146–10,436 | 15,320–15,899 |
| *XML* | | | | | |
| CARS measured | 4 | 36 | 68 | 163 | 316 |
| CARS predicted | 1–4 | 4–34 | 7–67 | 12–163 | 19–320 |
| ARS measured | 186 | 563 | 924 | 1,704 | 2,727 |
| ARS predicted | 111–114 | 484–514 | 760–820 | 1,389–1,539 | 2,191–2,491 |
| *SQL* | | | | | |
| CARS measured | 36 | 327 | 590 | 1,414 | 2,801 |
| CARS predicted | 6–187 | 31–1,834 | 51–3,656 | 91–9,104 | 136+ |
| ARS measured | – | – | – | – | – |
| ARS predicted | 52,595–52,775 | 106,264–108,067 | 115,001–118,606 | 122,038–131,051 | 125,065–143,091 |
| *JSON (grammar in prompt)* | | | | | |
| CARS measured | 3 | 37 | 66 | 153 | 297 |
| CARS predicted | 1 | 2 | 2 | 3 | 4 |
| ARS measured | 4,256 | – | – | – | – |
| ARS predicted | 5,109 | 23,758 | 34,550 | 52,576 | 67,910 |
| *XML (grammar in prompt)* | | | | | |
| CARS measured | 1 | 12 | 23 | 55 | 115 |
| CARS predicted | 1 | 2 | 3 | 3 | 4 |
| ARS measured | 1,403 | 7,150 | – | – | – |
| ARS predicted | 1,795 | 9,153 | 14,251 | 24,460 | 35,267 |
| *SQL (grammar in prompt)* | | | | | |
| CARS measured | 12 | 114 | 237 | 602 | 1,149 |
| CARS predicted | 2 | 5 | 8 | 15 | 23 |
| ARS measured | – | – | – | – | – |
| ARS predicted | 97,266 | 124,281 | 126,075 | 127,290 | 127,757 |

Table 5: Rejected LM calls before the $n$-th valid sample: means over five independent runs, and the prediction (Theorem 3.1) from the trie of a further CARS run; "–": the runs ended before their $n$-th valid sample.

| Benchmark | root $P(\varepsilon \mathrm{bad}_\varepsilon \mid \varepsilon)$ | non-root seen mass, by depth of $v$ 1–2 | 3–5 | 6–10 | 11–20 | $> 20$ | non-root seen mass, by $P(va)$ heavy | medium | light | unseen $/P(\mathcal{L})$ |
|---|---|---|---|---|---|---|---|---|---|---|
| SMILES (acrylates) | 0.001 | 0.16 | 0.07 | 0.25 | 0.51 | 0.00 | 0.00 | 0.83 | 0.17 | 2.43 |
| SyGuS SLIA | 0.979 | 0.00 | 0.00 | 0.34 | 0.04 | 0.61 | 0.93 | 0.06 | 0.01 | 0.07 |
| SyGuS BV4 | 1.000 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.41 | 0.50 | 0.08 | 0.51 |
| JSON | 1.000 | 0.75 | 0.10 | 0.04 | 0.06 | 0.04 | 0.00 | 0.34 | 0.66 | 5.79 |
| XML | 1.000 | 0.24 | 0.00 | 0.00 | 0.32 | 0.44 | 0.00 | 0.29 | 0.71 | 3.00 |
| SQL | 1.000 | 0.00 | 0.00 | 0.00 | 0.82 | 0.18 | 0.84 | 0.11 | 0.05 | 180.26 |
| JSON (grammar in prompt) | 1.000 | 0.71 | 0.02 | 0.12 | 0.09 | 0.06 | 0.00 | 0.37 | 0.63 | 0.00 |
| XML (grammar in prompt) | 1.000 | 0.90 | 0.00 | 0.00 | 0.03 | 0.08 | 0.00 | 0.31 | 0.69 | 0.00 |
| SQL (grammar in prompt) | 1.000 | 0.00 | 0.00 | 0.03 | 0.71 | 0.25 | 0.68 | 0.09 | 0.23 | 0.00 |

Table 6: Where the invalid mass sits: $P(\varepsilon \mathrm{bad}_\varepsilon \mid \varepsilon)$ at the root; the non-root mass the trie has seen, split by the depth of the prefix and by atom mass (heavy: $\geq P(\mathcal{L})$; medium: $\geq P(\mathcal{L})/100$; light: below); and the mass at prefixes never reached, in units of $P(\mathcal{L})$, i.e., the RS-rate calls per valid sample it costs.

| Benchmark | prefix $v$ | $P(v)$ | $P(v\mathrm{bad}_v \mid v)$ | $\lvert\mathrm{bad}_v\rvert$ | $\rho_v$ |
|---|---|---|---|---|---|
| SMILES (acrylates) | `CC(=C)C(=O)OC1=C` | 0.056 | 0.805 | 126,383 | 0.453 |
| | `CC` | 0.592 | 0.076 | 126,383 | 0.035 |
| | `CC(=C)C(=O)OCC1=C` | 0.048 | 0.823 | 126,383 | 0.501 |
| SyGuS SLIA | $\varepsilon$ | 1.000 | 0.979 | 128,255 | 0.000 |
| | `... 0 (str.indexof name " " 0` | 0.013 | 0.926 | 128,255 | 0.003 |
| | `...n f ((name String)) String` | 0.020 | 0.326 | 128,249 | 0.000 |
| SyGuS BV4 | $\varepsilon$ | 1.000 | 1.000 | 128,255 | 0.000 |
| | `... (BitVec 4))) (BitVec 4) #` | $1.6 \times 10^{-4}$ | 0.853 | 128,255 | 0.108 |
| | `...c 4))) (BitVec 4) (bvneg t` | $2.2 \times 10^{-5}$ | 0.806 | 128,255 | 0.526 |
| JSON | $\varepsilon$ | 1.000 | 1.000 | 128,255 | 0.000 |
| | `{"` | $4.6 \times 10^{-6}$ | 0.049 | 128,255 | 0.000 |
| | `{"name"` | $2.6 \times 10^{-8}$ | 1.000 | 128,229 | 0.962 |
| XML | $\varepsilon$ | 1.000 | 1.000 | 128,254 | 0.000 |
| | `<` | $1.5 \times 10^{-6}$ | 0.995 | 128,253 | 0.889 |
| | `...="1.0" encoding="UTF-8"?>` | $5.4 \times 10^{-5}$ | 0.021 | 127,827 | 0.000 |
| SQL | $\varepsilon$ | 1.000 | 1.000 | 128,255 | 0.000 |
| | `...urce $testdir/tester.tcl` | $1.9 \times 10^{-8}$ | 0.503 | 127,825 | 0.000 |
| | `...keys-1.0 { execsql {` | $6.2 \times 10^{-10}$ | 0.634 | 127,822 | 0.299 |
| JSON (grammar in prompt) | $\varepsilon$ | 1.000 | 1.000 | 128,255 | 0.000 |
| | `{"` | $2.9 \times 10^{-8}$ | 0.026 | 128,255 | 0.000 |
| | `{"name` | $2.8 \times 10^{-8}$ | 0.009 | 128,244 | 0.000 |
| XML (grammar in prompt) | $\varepsilon$ | 1.000 | 1.000 | 128,254 | 0.000 |
| | `<` | $1.0 \times 10^{-8}$ | 0.993 | 128,253 | 0.716 |
| | `...!ELEMENT price (#PCDATA)>` | $7.1 \times 10^{-10}$ | 0.270 | 127,827 | 0.973 |
| SQL (grammar in prompt) | $\varepsilon$ | 1.000 | 1.000 | 128,255 | 0.000 |
| | `...urce $testdir/tester.tcl` | $6.0 \times 10^{-11}$ | 0.095 | 127,825 | 0.000 |
| | `...keys-1.0 { execsql {` | $1.0 \times 10^{-12}$ | 0.154 | 127,822 | 0.803 |

Table 7: The three prefixes with the most invalid mass $P(v)P(v\mathrm{bad}_v \mid v)$ per task (long prefixes shown by their tail).

| $n$ | $\mathbb{E}\mathrm{Rej}_{\mathrm{CARS}}(n)$ measured | predicted | $\mathbb{E}\mathrm{Rej}_{\mathrm{ARS}}(n)$ measured | predicted |
|---|---|---|---|---|
| 5 | 0.8 | 1.0–1.1 | 2.9 | 3.0–3.1 |
| 10 | 1.1 | 1.4–1.6 | 5.7 | 5.7–5.9 |
| 20 | 1.7 | 1.8–2.2 | 10.6 | 10.3–10.8 |
| 40 | 2.8 | 2.2–3.2 | 18.3 | 17.7–18.7 |
| 60 | 3.7 | 2.5–4.0 | 25.2 | 23.9–25.4 |
| 80 | 4.3 | 2.8–4.8 | 31.4 | 29.3–31.3 |

The ranges bracket the unseen mass (0.015 here). Every ARS mean lies inside its range, and so does CARS from $n = 20$ on; at $n = 5$ and 10 CARS is about one standard error below, where the range is narrowest and one early rejection moves the mean. RS matched $n(1 - \hat{P}(\mathcal{L}))/\hat{P}(\mathcal{L})$ to within $0.6\%$ at $n = 100$. Two lessons carried over to the benchmarks. A length cap set close to the typical valid length truncates CARS's draws, and only CARS's, because pruning short invalid continuations

| Benchmark | | trials | mean $\mathrm{Rej}(100)$ | | std. dev. of $\mathrm{Rej}(100)$ | |
|---|---|---|---|---|---|---|
| | | | measured | predicted | measured | predicted |
| SMILES (acrylates) | CARS | 10 | 389 | 171–415 | 32.9 | 8.3 |
| SMILES (acrylates) | ARS | 5 | 676 | 460–703 | 39.4 | 30.8 |
| SyGuS SLIA | CARS | 10 | 15 | 11–19 | 2.8 | 1.8 |
| SyGuS SLIA | ARS | 5 | 891 | 776–783 | 26.6 | – |
| SyGuS BV4 | CARS | 10 | 77 | 39–91 | 7.0 | 4.1 |
| SyGuS BV4 | ARS | 5 | 3241 | 3,510–3,562 | 146.9 | – |
| JSON | CARS | 10 | 600 | 5–583 | 42.4 | 1.6 |
| XML | CARS | 10 | 316 | 19–320 | 30.4 | 3.4 |
| XML | ARS | 5 | 2727 | 2,191–2,491 | 69.4 | – |
| SQL | CARS | 10 | 2801 | 136–18,162 | 361.6 | 8.8 |
| JSON (grammar in prompt) | CARS | 10 | 297 | 4 | 46.0 | 1.7 |
| XML (grammar in prompt) | CARS | 10 | 115 | 4 | 15.1 | 1.4 |
| SQL (grammar in prompt) | CARS | 10 | 1149 | 23 | 77.4 | 3.8 |

Table 8: Rejected LM calls before the 100-th valid sample over five independent runs: mean and standard deviation, measured and predicted (Theorems A.3 and 3.1, from the seed-0 trie; "–": the ARS mixture is not tabulated for $P(\mathcal{L}) < 10^{-3}$).

pushes its proposal towards long expressions; at 16 tokens this was 5 of 21 rejections, at 32 it was $1.4\%$, and at the benchmark caps of 256–512 tokens it never happened. And the trie dump must keep every light atom: a first version that kept only the 64 heaviest per prefix under-predicted ARS by $15\%$, since the dropped atoms are exactly the ones ARS pays for.

**Predicting without a run.** Everything in Theorem 3.1 except $P(\mathcal{L})$ is local to a prefix: $P(v\mathrm{bad}_v \mid v)$, $|\mathrm{bad}_v|$ and the atom masses take one forward pass and one oracle query at $v$, and $P(v)$ is the product of the conditionals along the path. Expanding grammatical prefixes in decreasing $P(v)$ down to a threshold $t$ therefore enumerates exactly the prefixes $R_t$ sums over, and the prefixes below $t$ need no enumeration: within $k \approx 1/t$ calls CARS is not expected to reach them (Theorem 4.1), so Theorem 4.1 charges their mass in full, as $U_t = 1 - P(\mathcal{L}) - R_t$; the lifetime CARS cost $\sum_v P(v\mathrm{bad}_v \mid v)$ is available before any sample is drawn. This is how the last four columns of Table 9 are computed. What does need sampling is $P(\mathcal{L})$, a sum over the whole language, which sets the scale of every cost: ARS's rate per atom, RS's $1/P(\mathcal{L})$, the ceiling on the speedup and, through $\rho_v$, the CARS rate. So *whether* CARS will help is answerable from an enumeration; *how many* calls it takes needs an estimate of $P(\mathcal{L})$, from a short run.

## H What is machine-checked

Part of the analysis is formalized in Lean 4. This appendix says exactly which part, so that nothing in the body has to be read as carrying more assurance than it does.

**Summary.** *Checked:* the per-atom survival bounds, the identity expressing $\mathbb{E}[p_\varepsilon]$ as a sum over atoms, the two threshold bounds behind Theorem 4.1 and the two envelopes they use, the exact per-atom rates in the form of matching bounds at every fixed number of draws, exactness and monotonicity of the samplers, and the consequences of Theorem A.4—that $p_\varepsilon$ under CARS is below $p_\varepsilon$ under any dominated strategy, in expectation and in distribution, at every iteration. *Not checked:* the stream coupling of Section 2 and everything proved through it, namely Theorem 3.1, Corollaries C.1 to C.2, Theorem A.3, and the pathwise form of Theorem A.4. The reason is given at the end of this appendix; it is structural, not a matter of effort. The seven files listed below total about 3,400 lines, about 4,500 with the files they import. It contains no `sorry` and introduces no axioms: every statement below depends only on `propext`, `Classical.choice` and `Quot.sound`, the three axioms of Lean's core logic.

**The model.** The files `Cars.lean`, `CarsSpeedup.lean`, `CarsAggregate.lean`, `CarsBadTokens.lean`, `CarsEnvelope.lean`, `CarsExact.lean` and `CarsDominance.lean` in `LeanTheorems/` prove the following in the abstract model

| | | | calls(100), measured | | predicted | | calls ARS/CARS | | enumeration, no sampling | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Benchmark | model | $\hat{P}(\mathcal{L})$ | CARS | ARS | CARS | ARS | meas. | pred. | $R_{P(\mathcal{L})/100}$ | $\sum P(v\mathrm{bad}_v \mid v)$ | light | bound |
| SMILES | Llama-8B | 0.049 | 489 | 776 | 271–515 | 560–803 | 1.6 | 2.1 | 0.823 | 163.8 | 0.217 | 1.5 |
| | Llama-8B base | $1.0 \times 10^{-4}$ | 890 | 7,979 | 351–1,063 | 8,712–9,424 | 9.0 | 24.8 | 0.999 | 222.5 | 0.006 | 6.8 |
| | Qwen-7B | $3.0 \times 10^{-3}$ | 148 | 191 | 130–155 | 169–194 | 1.3 | 1.3 | 0.996 | 24.4 | 0.002 | 1.2 |
| | Qwen-14B | 0.085 | 147 | 187 | 137–154 | 169–187 | 1.3 | 1.2 | 0.907 | 32.9 | 0.029 | 1.2 |
| | Qwen-Coder-7B | 0.072 | 251 | 376 | 171–259 | 288–376 | 1.5 | 1.7 | 0.860 | 68.9 | 0.128 | 1.4 |
| | Qwen-Math-7B | $< 7.0 \times 10^{-7}$ | – | – | 1,015–49,360 | $1.5 \times 10^{6}$–$1.5 \times 10^{6}$ | – | 1,453.9 | 1.000 | 3,143.8 | 0.008 | 61.5 |
| | phi-4 | 0.087 | 249 | 361 | 170–263 | 273–366 | 1.4 | 1.6 | 0.835 | 70.2 | 0.143 | 1.4 |
| | Llama-70B | 0.282 | 168 | 201 | 137–181 | 168–211 | 1.2 | 1.2 | 0.577 | 34.9 | 0.206 | 1.2 |
| SLIA | Llama-8B | $8.5 \times 10^{-4}$ | 115 | 991 | 111–119 | 876–883 | 8.6 | 7.9 | 0.999 | 10.9 | 0.005 | 6.2 |
| | Llama-8B base | $9.0 \times 10^{-3}$ | 144 | 340 | 128–150 | 327–350 | 2.4 | 2.6 | 0.988 | 24.5 | 0.017 | 2.2 |
| | Qwen-7B | $5.0 \times 10^{-8}$ | 156 | $5{,}704^{(n=12)}$ | 134–3,977 | 20,424–24,266 | $179.4^{(n=12)}$ | 152.5 | 1.000 | 33.6 | 0.000 | $\infty$ |
| | Qwen-14B | $6.3 \times 10^{-6}$ | 105 | 222 | 105–153 | 234–283 | 2.1 | 2.2 | 1.000 | 3.0 | 0.000 | 1.5 |
| | Qwen-Coder-7B | 0.891 | 100 | 106 | 100–101 | 106–107 | 1.1 | 1.1 | 0.101 | 0.1 | 0.042 | 1.0 |
| | Qwen-Math-7B | $2.9 \times 10^{-24}$ | 328 | – | 299–$5.0 \times 10^{19}$ | $4.9 \times 10^{6}$–$5.0 \times 10^{19}$ | – | 16,369.6 | 1.000 | 195.3 | 0.000 | 1.0 |
| | phi-4 | $9.7 \times 10^{-19}$ | 316 | – | 241–$1.3 \times 10^{14}$ | $1.0 \times 10^{5}$–$1.3 \times 10^{14}$ | – | 423.2 | 1.000 | 124.1 | 0.000 | 1.0 |
| | Llama-70B | $1.3 \times 10^{-5}$ | 101 | 6,887 | 102–111 | 6,778–6,787 | 68.2 | 66.4 | – | – | – | – |
| BV4 | Llama-8B | $2.7 \times 10^{-5}$ | 177 | 3,341 | 139–195 | 3,610–3,666 | 18.9 | 25.9 | 1.000 | 37.5 | 0.001 | 17.1 |
| | Llama-8B base | $4.1 \times 10^{-3}$ | 560 | 1,551 | 349–642 | 1,494–1,787 | 2.8 | 4.3 | 0.983 | 216.2 | 0.045 | 2.9 |
| | Qwen-7B | $3.6 \times 10^{-6}$ | 1,044 | 6,020 | 612–1,661 | 5,398–6,447 | 5.8 | 8.8 | 1.000 | 510.0 | 0.000 | 7.0 |
| | Qwen-14B | $1.5 \times 10^{-6}$ | 474 | 1,038 | 334–618 | 841–1,126 | 2.2 | 2.5 | 1.000 | 227.5 | 0.000 | 1.8 |
| | Qwen-Coder-7B | 0.641 | 113 | 129 | 106–117 | 124–134 | 1.1 | 1.2 | 0.276 | 4.4 | 0.162 | 1.1 |
| | Qwen-Math-7B | $4.0 \times 10^{-31}$ | 512 | – | 443–$2.8 \times 10^{26}$ | $6.2 \times 10^{6}$–$2.8 \times 10^{26}$ | – | 13,893.5 | 1.000 | 350.9 | 0.000 | $\infty$ |
| | phi-4 | $2.0 \times 10^{-14}$ | 400 | – | 293–$5.6 \times 10^{9}$ | 94,553–$5.6 \times 10^{9}$ | – | 322.4 | 1.000 | 182.9 | 0.000 | $\infty$ |
| JSON | Llama-8B | $6.5 \times 10^{-7}$ | 700 | $6{,}737^{(n=26)}$ | 105–858 | 15,420–16,174 | $36.2^{(n=26)}$ | 147.2 | 1.000 | 3.9 | 0.000 | 15.7 |
| | Llama-8B base | $1.1 \times 10^{-7}$ | 1,302 | $5{,}979^{(n=12)}$ | 153–16,350 | 19,742–35,940 | $5.2^{(n=12)}$ | 128.9 | 1.000 | 21.1 | 0.000 | 1.0 |
| | Qwen-7B | $4.6 \times 10^{-10}$ | 274 | 5,638 | 101–$2.5 \times 10^{5}$ | 6,140–$2.5 \times 10^{5}$ | 20.6 | 60.7 | 1.000 | 1.0 | 0.000 | 1.0 |
| | Qwen-14B | $3.2 \times 10^{-11}$ | 1,686 | 6,418 | 123–$3.5 \times 10^{6}$ | 3,118–$3.5 \times 10^{6}$ | 3.8 | 25.4 | 1.000 | 7.2 | 0.000 | 1.0 |
| | Qwen-Coder-7B | $7.8 \times 10^{-6}$ | 227 | 2,350 | 102–243 | 2,134–2,276 | 10.3 | 21.0 | 1.000 | 1.1 | 0.000 | 8.0 |
| | Qwen-Math-7B | $< 3.9 \times 10^{-9}$ | – | – | 159–$2.3 \times 10^{5}$ | $1.5 \times 10^{5}$–$3.8 \times 10^{5}$ | – | 971.9 | 1.000 | 310.0 | 0.000 | 1.0 |
| | phi-4 | $2.3 \times 10^{-13}$ | 667 | $7{,}055^{(n=15)}$ | 118–$4.9 \times 10^{8}$ | 9,854–$4.9 \times 10^{8}$ | – | 83.8 | 1.000 | 1.9 | 0.000 | $\infty$ |
| XML | Llama-8B | $1.2 \times 10^{-5}$ | 416 | 2,827 | 119–429 | 2,291–2,600 | 6.8 | 19.2 | 1.000 | 7.4 | 0.000 | 4.7 |
| | Llama-8B base | $3.8 \times 10^{-6}$ | 1,681 | $4{,}049^{(n=1)}$ | 218–5,116 | 51,233–56,131 | $57.0^{(n=1)}$ | 234.7 | 1.000 | 82.6 | 0.001 | 4.7 |
| | Qwen-7B | $2.9 \times 10^{-5}$ | 1,498 | 1,714 | 258–1,562 | 406–1,710 | 1.1 | 1.6 | 1.000 | 75.7 | 0.000 | 1.0 |
| | Qwen-14B | $2.6 \times 10^{-4}$ | 204 | 204 | 108–207 | 119–219 | 1.0 | 1.1 | 0.999 | 5.5 | 0.000 | 1.0 |
| | Qwen-Coder-7B | 0.080 | 628 | 641 | 144–568 | 169–593 | 1.0 | 1.2 | 0.575 | 35.6 | 0.050 | 1.0 |
| | Qwen-Math-7B | $< 4.6 \times 10^{-7}$ | – | – | 152–$2.0 \times 10^{5}$ | $2.6 \times 10^{5}$–$4.6 \times 10^{5}$ | – | 1,732.9 | 0.999 | 185.9 | 0.000 | 1.0 |
| | phi-4 | 0.015 | 131 | 140 | 103–143 | 111–151 | 1.1 | 1.1 | 0.979 | 2.4 | 0.001 | 1.0 |
| SQL | Llama-8B | $2.6 \times 10^{-10}$ | 2,901 | – | 236–$4.5 \times 10^{5}$ | $1.3 \times 10^{5}$–$5.7 \times 10^{5}$ | – | 529.3 | 1.000 | 52.8 | 0.000 | 1.0 |
| | Llama-8B base | 0.013 | 1,509 | 6,672 | 194–6,627 | 579–7,012 | 4.4 | 3.0 | 0.143 | 47.1 | 0.045 | 1.0 |
| | Qwen-7B | $5.5 \times 10^{-4}$ | 1,367 | 12,660 | 325–21,917 | 687–22,279 | 9.3 | 2.1 | 0.914 | 22.3 | 0.001 | 1.0 |
| | Qwen-14B | $4.3 \times 10^{-4}$ | 1,810 | 2,103 | 143–1,923 | 167–1,947 | 1.2 | 1.2 | 0.994 | 11.7 | 0.000 | 1.0 |
| | Qwen-Coder-7B | 0.129 | 565 | 637 | 118–554 | 141–577 | 1.1 | 1.2 | 0.297 | 10.6 | 0.031 | 1.0 |
| | Qwen-Math-7B | $< 9.4 \times 10^{-8}$ | – | – | 990–$1.9 \times 10^{5}$ | $2.7 \times 10^{6}$–$2.8 \times 10^{6}$ | – | 2,678.3 | 1.000 | 1,914.7 | 0.001 | 4.3 |
| | phi-4 | $2.7 \times 10^{-5}$ | 1,749 | 2,820 | 203–3,248 | 419–3,463 | 1.6 | 2.1 | 0.999 | 26.5 | 0.000 | 1.0 |
| JSON+g | Llama-8B | $5.8 \times 10^{-9}$ | 397 | – | 104–19,748 | 68,010–87,653 | – | 652.3 | 1.000 | 3.2 | 0.000 | $\infty$ |
| | Qwen-7B | $2.6 \times 10^{-9}$ | 148 | 1,696 | 101–43,716 | 1,746–45,361 | 11.4 | 17.3 | 1.000 | 1.0 | 0.000 | $\infty$ |
| XML+g | Llama-8B | $2.6 \times 10^{-8}$ | 215 | $4{,}116^{(n=2)}$ | 104–4,407 | 35,367–39,670 | $980.0^{(n=2)}$ | 340.0 | 1.000 | 3.0 | 0.000 | 50.4 |
| | Qwen-7B | $7.3 \times 10^{-3}$ | 659 | 699 | 166–654 | 180–668 | 1.1 | 1.1 | 0.959 | 47.5 | 0.004 | 1.0 |
| SQL+g | Llama-8B | $4.1 \times 10^{-12}$ | 1,249 | – | 123–$2.8 \times 10^{7}$ | $1.3 \times 10^{5}$–$2.8 \times 10^{7}$ | – | 1,042.5 | 1.000 | 11.7 | 0.000 | 1.0 |
| | Qwen-7B | $1.0 \times 10^{-3}$ | 585 | $5{,}631^{(n=10)}$ | 269–49,441 | 665–49,837 | – | 2.5 | 0.535 | 24.6 | 0.002 | 1.0 |
| arith | codegen-350M | 0.605 | 105 | 138 | 103–105 | 134–136 | 1.3 | 1.3 | – | – | – | – |

Table 9: The model axis: six benchmarks against seven checkpoints, with Llama-3.1-70B added on SMILES and SLIA and the three grammar-in-prompt variants against Llama-8B and Qwen-7B. *calls(100)* are LM calls to the 100th valid sample, measured (means over seeds 1–9 for CARS and 1–4 for ARS; a superscript $(n)$ marks a mean at the largest $n$ all ARS runs reached) and predicted by Theorem 3.1 from the seed-0 trie, as in Table 1. *calls ARS/CARS* is the ratio of the two costs, measured and predicted; it is not $\mathrm{Speedup}(100)$ of Table 4, which compares $p_\varepsilon$ after 100 calls. The last four columns come from *enumeration alone*, with no sampling: the invalid mass at prefixes with $P(v) \geq P(\mathcal{L})/100$, the lifetime CARS cost $\sum P(v\mathrm{bad}_v \mid v)$ over those prefixes (Corollary 3.2), the light mass, and the resulting bound on the speedup at those thresholds. "–": the run did not reach 100 valid samples within its budget.

of the formalization: complete sequences are an arbitrary type with a probability mass function $P$, a strategy is a map $w \mapsto \textsc{Invalid}(w, \mathcal{L})$ with $\textsc{Invalid}(w, \mathcal{L})\Sigma^*\$ \cap \mathcal{L} = \emptyset$, and an atomic rule is a family of disjoint atoms $A_i$ covering $\mathcal{L}^c$ with discovery sets $D_i$. We write $\mathrm{Surv}_A(k)$ for the probability that $A$ is still not pruned after $k$ draws.

| Lean theorem | hypothesis | conclusion |
|---|---|---|
| `surv_le_pow` | every draw in $D$ prunes $A$ | $\mathrm{Surv}_A(k) \leq (1 - P(D \cap \mathcal{L}))^k$ |
| `surv_le_pow_of_disjoint` | … and draws outside $D$ exclude nothing in $D$ | $\mathrm{Surv}_A(k) \leq (1 - P(D))^k$ |
| `surv_biUnion_le_pow` | atomic rule; $D_0 \subseteq D_i$ for $i \in J$; as above for $D_0$ | $\mathrm{Surv}_{\bigcup_J A_i}(k) \leq (1 - P(D_0))^k$ |
| `pow_le_surv` | only draws in $D$ prune $A$ | $\mathrm{Surv}_A(k) \geq (1 - P(D)/P(\mathcal{L}))^k$ |
| `ars_card_le_length` | $D_i = A_i$; $n$ nonempty atoms pruned after $k$ draws | $n \leq k$, pathwise |
| `accBefore_le_pow` | every draw in $D$ prunes $A$ | $\mathbb{P}[\geq n$ accepts before $A$ pruned$] \leq (1 - P(D \cap \mathcal{L})/P(\mathcal{L}))^n$ |
| `prunedBefore_le_of_subset` | $A \subseteq B$; only draws in $A$ prune $A$ | $\mathbb{P}[B$ pruned before $n$-th accept$] \leq 1 - (1 - m_A/(m_A + P(\mathcal{L})))^n$ |
| `accBefore_le_pow_exact` | every draw in $D$ prunes $A$; draws outside $D$ exclude nothing in $D$ | $\mathbb{P}[\geq n$ accepts before $A$ pruned$] \leq (1 - q_D)^n$, $q_D = \frac{P(D)}{P(D)+P(\mathcal{L}\setminus D)}$ |
| `prunedBefore_le_general` | only draws in $D$ prune $A$ | $\mathbb{P}[A$ pruned before $n$-th accept$] \leq 1 - (1 - q_D)^n$ |
| `expected_pε_eq_atoms` | atomic rule | identity (1) |

| Lean theorem | hypothesis | conclusion |
|---|---|---|
| `expected_pε_le_atoms_strong` | atomic rule; outside draws exclude nothing in $D_i$ | $\mathbb{E}p_\varepsilon(\mathcal{W}_k) \le P(\mathcal{L}) + \sum_i m_i(1-P(D_i))^k$ |
| `expected_pε_ge_atoms` | $D_i = A_i$ | $\mathbb{E}p_\varepsilon(\mathcal{W}_k) \ge P(\mathcal{L}) + \sum_i m_i(1 - \min\{1, m_i/P(\mathcal{L})\})^k$ |
| `expected_pε_le_threshold`, `_le_heavy` | as `_le_atoms_strong`; $t \in [0,1]$ | $\mathbb{E}p_\varepsilon(\mathcal{W}_k) \le 1-(1-(1-t)^k)R_t$ |
| `expected_pε_ge_threshold`, `_ge_light` | $D_i = A_i$; $s \in [0, P(\mathcal{L})]$ | $\mathbb{E}p_\varepsilon(\mathcal{W}_k) \ge P(\mathcal{L}) + (1-s/P(\mathcal{L}))^k\Lambda_s$ |
| `surv_le_prod_env` | every draw in $D$ prunes $A$; outside draws exclude nothing in $D$; $p_\varepsilon(\mathcal{W}_i) \le c_i$ on every run | $\mathrm{Surv}_A(k) \le \prod_{i<k}(1-P(D)/c_i)_+$ |
| `prod_le_surv_env` | only draws in $D$ prune $A$; $p_\varepsilon(\mathcal{W}_i) \ge \ell_i$ on every run; $P(D) \le \ell_i$ | $\mathrm{Surv}_A(k) \ge \prod_{i<k}(1-P(D)/\ell_i)$ |
| `chainEnv_step`, `chainEnv_bound` | $D_0 = \Sigma^*\$ \supseteq D_1 \supseteq \dots$; every draw in $D_j$ prunes $B_j$; $D_j \setminus B_j \subseteq D_{j+1}$ | after $i$ draws $B_0, \dots, B_{i-1}$ are pruned and $p_\varepsilon \le 1 - P(B_0 \cup \dots \cup B_{i-1})$ |
| `arsEnv_step`, `arsEnv_bound` | $D_i = A_i$; any $i$ atoms have mass $\le M_i$ | after $i$ draws $p_\varepsilon \ge 1 - M_i$ |
| `expected_pε_le_heavy_chain`, `_ge_light_ars` | the two envelopes above | Theorem B.1 with one edge, $t$ and $s$ |
| `expected_pε_le_rounded_env`, `_ge_rounded_env` | any $\tau_i \le P(D_i)$ (resp. any $\sigma_i \ge m_i$ with $\sigma_i \le \ell_j$ on a set of kept atoms) | the envelope bounds with $\tau_i$ (resp. $\sigma_i$) in place of the true masses |
| `expected_pε_le_bins_env`, `_ge_bins_env` | the roundings take values in a finite set of edges | Theorem 4.1 on a grid: the sums regrouped by edge (`tsum_bins_eq`) |
| `cars_soundness` | $\mathcal{W}$ disjoint from $\mathcal{L}$, $p_\varepsilon > 0$ | the accepted draws have law $P^{\mathcal{L}}$: exactness |
| `cars_monotonicity` | $\mathcal{W} \subseteq \mathcal{W}'$ | the acceptance rate does not decrease |
| `expected_pε_le` | $S_2$ dominates $S_1$ (`Dominates`: $S_1 w \subseteq S_2 w$, and $S_2$ closed under what $S_1$ learns) | $\mathbb{E}p_\varepsilon$ under $S_2$ is at most under $S_1$, at every $k$ |
| `stochastic_dominance` | as above | $\mathbb{P}[p_\varepsilon(\mathcal{W}_k) > t]$ under $S_2$ is at most under $S_1$, for every $t$ and $k$ |
| `expected_rejection_le` | as above | the expected rejection probability under $S_2$ is at most under $S_1$, at every $k$ |

The exact rates of Theorem A.2(b),(c) are checked as two bounds at a fixed number of draws with one and the same rate $q_D$: for every number of draws $k$, the probability of $n$ accepts with $A$ unpruned is at most $(1-q_D)^n$ (`accBefore_le_pow_exact`), and the probability of pruning before the $n$-th accept is at most $1-(1-q_D)^n$ (`prunedBefore_le_general`); with $D = v\Sigma^*\$$ this $1-q_D$ is $\rho_v$, with $D = A_{v,a}$ it is $P(\mathcal{L})/(P(\mathcal{L})+P(va))$ (`rate_atom`, `one_sub_rate`). Since fewer than $n$ accepts in $k$ draws becomes impossible as $k \to \infty$, the two bounds pin the probability of surviving the whole run; that passage to the limit is the one step left informal. Theorem A.4 is thus machine-checked in its consequences—$p_\varepsilon$ under CARS is below $p_\varepsilon$ under any dominated strategy in expectation and in distribution, at every iteration—but not in the pathwise form stated in Section 3, which quantifies over a fixed sample sequence rather than over the draw-count recursion. Theorem A.3 is not in Lean, and the obstacle is the shape of the formalization rather than the difficulty of the proof. In the model above a strategy's cost is defined by recursion on the number of draws—the expected value after $k$ draws is a sum over the next draw of the expected value after $k-1$—and that recursion never names a probability space. There are no random variables in it, so there is nothing of which independence can be asserted, and the content of Theorem A.3(a) is precisely that the per-prefix indicators *are* independent. Stating it would mean building the run as a measure on infinite sequences, defining the per-prefix token streams, and proving that they induce the law the recursion computes; the recursion was chosen in the first place to avoid exactly that machinery. The same obstacle applies to the pathwise form of Theorem A.4, which quantifies over a fixed sample sequence.

Beyond the list in the summary, what is argued in prose rather than in Lean is: that CARS is the atomic rule with the sets $\mathrm{atoms}(v)$ as atoms and $D_v = \{w : v \prec w\}$ as discovery sets, that ARS is the one with $D_{v,a} = A_{v,a}$, and that draws outside $D_v$ exclude nothing inside $D_v$ and that the forced chain satisfies the chain hypotheses, $D_j = F_j\Sigma^*\$$ and $D_j \setminus \mathrm{atoms}(F_j) = D_{j+1}$ when $F_{j+1}$ is the unique grammatical child of $F_j$ (all short consequences of prefix-closedness of $\mathrm{prefix}(\mathcal{L})$; formalizing them needs sequences as lists); the harmonic factor $H_{|\mathrm{bad}_v|}$ in the exact ARS cost of a node (Section D); and the "one spine level per rejected draw" observation. The running example's table and simulation are computed by the script `example.py` in the supplementary material, not in Lean.