Tractable Constrained Generation with LL(1) Grammars

Jaron Maene[1] **Guy Van den Broeck**[2]

[1]KU Leuven, Dept. of Computer Science, B-3000 Leuven, Belgium
[2]Department of Computer Science, University of California, Los Angeles, USA

Abstract

Steering a language model towards satisfying a constraint requires marginalizing the probability that a future continuation will be valid. Hidden Markov models have proven effective as tractable surrogates for such marginalization, but existing methods only handle constraints expressible as regular languages. We extend this line of work to LL(1) grammars, a widely used subset of context-free grammars. Unfortunately, even with a tractable surrogate, constrained generation with unambiguous grammars is still cubic time in the sequence length. We hence develop **CTRL-CFG**, a linear-time approximation that marginalizes only over a fixed look-ahead window. We demonstrate the effectiveness of CTRL-CFG on chemistry-grammar conditioned molecule generation.

1 INTRODUCTION

Large language models (LLMs) are widely used for structured generation tasks such as code generation, formal theorem proving, and tool calling with JSON APIs. In all of these settings, the LLM-generated text must conform to a *context-free grammar* (CFG). Unfortunately, LLMs are known to produce syntactically invalid outputs in practice [Geng et al., 2025, Zuo et al., 2025], and theoretical results have indicated that this is a fundamental limitation of the transformer architecture [Strobl et al., 2024].

Theorem 1 (Informal)**.** *Let* L *be a nontrivial context-free language. Then there does not exist a bounded-depth transformer* T *such that* $p_T(x) > 0$ *if and only if* $x \in \mathsf{L}$.

So sampling from a transformer inevitably generates ungrammatical outputs. *Constrained generation* solves this problem by conditioning the next-token distribution of the LLM $p(x_t \mid x_{1:t})$ on a context-free language L. By Bayes' rule, we get

$$p(x_t \mid x_{1:t}, x \in \mathsf{L}) \propto p(x_t \mid x_{1:t})\, p(x \in \mathsf{L} \mid x_{1:t+1}). \quad (1)$$

The second term $p(x \in \mathsf{L} \mid x_{1:t+1})$, the probability that the prefix $x_{1:t+1}$ will eventually be completed to a string x in L, is intractable. For this reason, practical implementations often resort to *greedy token masking*, by replacing the membership probability with the indicator $[\![\exists y : x_{1:t+1} y \in \mathsf{L}]\!]$ that $x_{1:t+1}$ can still be extended to a grammatical string [Scholak et al., 2021, Beurer-Kellner et al., 2024]. Although efficient, greedy token masking is a myopic approximation: the existence of a continuation says nothing about its probability.

At the other extreme, *sampling-based estimators* based on rejection sampling [Park et al., 2024, Melcer et al., 2026], sequential Monte Carlo [Lew et al., 2023, Loula et al., 2025, Lipkin et al., 2025], and MCMC [Gonzalez et al., 2025] approach the true posterior of (1). However, these methods are expensive as they require multiple forward passes through the language model.

Zhang et al. [2024] proposed a middle ground for constrained decoding: *compute the membership term of* (1) *on a tractable surrogate* on which the marginalization can be computed efficiently. Their construction is restricted to regular languages, however, while many practical languages are inherently context-free. Hence, **we extend the surrogate-model approach to context-free grammars.**

Unlike for regular languages, constrained generation with a tractable surrogate still takes cubic time for unambiguous CFGs [Marzouk and de La Higuera, 2022, Zhang et al., 2025]. We hence propose a *look-ahead window* upper bound that computes the membership probability over the next k tokens (Section 4). For LL(1) grammars, this allows constrained decoding in linear time. An empirical evaluation on grammar-conditioned molecule generation (Section 5) shows that the approximation improves over both the greedy per-token grammar mask and sampling-based baselines.

2 PRELIMINARIES

Grammars. We fix a finite vocabulary Σ. We write $x_{1:n} = x_1 \dots x_{n-1}$ for a sequence of length $n-1$. In particular, $x_{i:i} = \epsilon$ and $x_{i:i+1} = x_i$. The concatenation of two sequences x and y is denoted xy. A *language* is a set of sequences. We write Σ^n for the set of all length-n sequences over Σ and Σ^* for all finite sequences over Σ, including the empty sequence ϵ. The *Brzozowski derivative* $x^{-1}\mathsf{L}$ is the language of all possible continuations of a prefix x:

$$x^{-1}\mathsf{L} = \{y \in \Sigma^* \mid xy \in \mathsf{L}\} \quad \text{with } x \in \Sigma^*.$$

Definition 1 (Context-free grammar)**.** *A* context-free grammar $G = (\Sigma, \mathcal{N}, S, \mathcal{R})$ *consists of a set of terminals Σ, a set of nonterminals $\mathcal{N}$, a start symbol $S \in \mathcal{N}$, and a set of production rules $\mathcal{R} \subseteq \mathcal{N} \times (\Sigma \cup \mathcal{N})^*$. The leftmost rewrite relation $\Rightarrow$ of a grammar is defined as*

$$\{xAy \Rightarrow x\beta y \mid (A \to \beta) \in \mathcal{R},\ x \in \Sigma^*,\ y \in (\Sigma \cup \mathcal{N})^*\}.$$

The language of G is $\mathsf{L}(G) \coloneqq \{x \in \Sigma^* \mid S \overset{*}{\Rightarrow} x\}$, where $\overset{*}{\Rightarrow}$ is the reflexive-transitive closure of $\Rightarrow$. A grammar G is *unambiguous* when for each sequence $x \in \mathsf{L}(G)$, the derivation $S \overset{*}{\Rightarrow} x$ is unique.

We further assume grammars to be in *canonical binary form* (CBF). As explained in Appendix B, conversion into CBF does not reduce expressivity and has only linear overhead.

Definition 2 (Canonical binary form)**.** *A grammar G is in canonical binary form, if all rules in $\mathcal{R}$ are of the form $A \to \epsilon$, $A \to \sigma$, $A \to B$, or $A \to BC$, with $\sigma \in \Sigma$.*

A grammar is moreover in LL(1) when we know which production to apply by only looking at the next symbol. It follows that all LL(1) grammars are unambiguous.

Definition 3 (LL(1))**.** *For each $\alpha \in (\Sigma \cup \mathcal{N})^*$, the set* $\mathsf{first}(\alpha) \subseteq \Sigma \cup \{\epsilon\}$ *contains the terminals that can begin a derivation of α, together with ϵ if α derives it.*

$$\mathsf{first}(\alpha) = \{a \in \Sigma \mid \exists \beta \in (\Sigma \cup \mathcal{N})^* \colon \alpha \overset{*}{\Rightarrow} a\beta\} \cup \{\epsilon \mid \alpha \overset{*}{\Rightarrow} \epsilon\}$$

For each $A \in \mathcal{N}$, the set $\mathsf{follow}(A) \subseteq \Sigma$ *contains the terminals that can immediately follow A.*

$$\mathsf{follow}(A) = \{a \in \Sigma \mid \exists \beta_1, \beta_2 \in (\Sigma \cup \mathcal{N})^* \colon S \overset{*}{\Rightarrow} \beta_1 A a \beta_2\}$$

A grammar is LL(1) *when every pair of distinct productions $A \to \alpha$ and $A \to \beta$ satisfies (1)* $\mathsf{first}(\alpha) \cap \mathsf{first}(\beta) = \varnothing$*, and (2) if $\epsilon \in \mathsf{first}(\alpha)$ then* $\mathsf{first}(\beta) \cap \mathsf{follow}(A) = \varnothing$.

Hidden Markov models. An HMM defines a joint distribution over a sequence $x_{1:n}$ and latent variables $z_{1:n}$ by assuming the Markov property: x_t conditioned on the latent z_t is independent of any past states.

$$p(x_{1:n}, z_{1:n}) = p(z_1)\, p(x_1 \mid z_1) \prod_{i=2}^{n-1} p(z_i \mid z_{i-1})\, p(x_i \mid z_i).$$

This Markov property lets us iteratively compute the marginal $p(x_{1:n})$ in $O(nh^2)$ via the forward algorithm [Rabiner and Juang, 1986], where h is the number of states for each latent variable z_i.

$$\begin{aligned} p(x_{1:t+1}, z_t) &= p(x_t \mid z_t)\, p(x_{1:t}, z_t) \\ &= p(x_t \mid z_t) \sum_{z_{t-1}} p(z_t \mid z_{t-1})\, p(x_{1:t}, z_{t-1}) \end{aligned}$$

3 EXACT CFG MARGINALIZATION

As shown in (1), to generate a string $x_{1:n}$ of length $n-1$, we need to compute the probability that $x_{1:n}$ is part of the language $\mathsf{L}(G)$. For now, we ignore the prefix $x_{1:t}$.

$$p_{\mathsf{LLM}}(x_{1:n} \in \mathsf{L}(G)) = \mathbb{E}_{x_{1:n} \sim p_{\mathsf{LLM}}} [\![S \overset{*}{\Rightarrow} x_{1:n}]\!].$$

As the above expectation is intractable, Zhang et al. [2024] proposed to substitute the LLM distribution p_{LLM} with an HMM distribution p_{HMM}. For regular languages, the marginal can then be computed in linear time. On CFGs it unfortunately remains hard in general.

Theorem 2 (Bertoni et al. 1991)**.** *Given a context-free grammar G, computing $p_{\mathsf{HMM}}(x_{1:n} \in \mathsf{L}(G))$ is #P-hard.*

On the more restricted class of unambiguous grammars, however, marginalization is known to be polytime.

Theorem 3 (Marzouk and de La Higuera 2022)**.** *Given an unambiguous context-free grammar G with rules $\mathcal{R}$ and an HMM with h states, computing $p_{\mathsf{HMM}}(x_{1:n} \in \mathsf{L}(G))$ takes $O(n^2 h^3 |\mathcal{R}|)$ time and $O(nh^2|\mathcal{N}|)$ space.*

This complexity is achieved with a CYK-style chart recursion. The pseudo-code can be found in Appendix C. For each nonterminal $A \in \mathcal{N}$ and $1 \le \ell \le n$, we compute the joint probability $p(A \overset{*}{\Rightarrow} x_{1:\ell},\ z_\ell \mid z_1)$ that the span $x_{1:\ell}$ is derivable from A while the latent walks from z_1 to z_ℓ. The base cases handle the epsilon and terminal productions:

$$p(A \overset{*}{\Rightarrow} x_{1:1}, z_1 \mid z_1') = [\![A \overset{*}{\Rightarrow} \epsilon]\!]\, [\![z_1 = z_1']\!]$$

$$\text{and} \quad p(A \overset{*}{\Rightarrow} x_{1:2},\ z_2 \mid z_1) = \sum_{A \to \sigma} p(\sigma \mid z_1)\, p(z_2 \mid z_1).$$

In the inductive case ($\ell > 2$), we aggregate over the binary rules $A \to BC$ that derive A and the position ℓ' where B and C are split. Unambiguity of the grammar guarantees that at most one (production, split) pair derives a given $x_{1:\ell}$, meaning we can sum the mutually exclusive probabilities.

$$p(A \overset{*}{\Rightarrow} x_{1:\ell},\ z_\ell \mid z_1) = \sum_{\substack{A \to BC \\ \ell' \in \{1 \dots \ell\}}} \sum_{z_{\ell'}} p(B \overset{*}{\Rightarrow} x_{1:\ell'}, z_{\ell'} \mid z_1)\, p(C \overset{*}{\Rightarrow} x_{\ell':\ell}, z_\ell \mid z_{\ell'})$$

Unary rules are handled similarly, see Appendix C. As only the length of a span matters in an HMM and not its absolute position in the sequence[1], we use $C \overset{*}{\Rightarrow} x_{1:\ell-\ell'+1}$ instead of $C \overset{*}{\Rightarrow} x_{\ell':\ell}$. Finally, we obtain the target probability by marginalizing away the boundary HMM states:

$$p_{\mathsf{HMM}}(x_{1:n} \in \mathsf{L}(G)) = \sum_{z_1, z_n} p(z_1)\, p(S \overset{*}{\Rightarrow} x_{1:n}, z_n \mid z_1).$$

4 CTRL-CFG: WINDOW UPPER BOUND

For constrained decoding, we need to evaluate the membership probability *at every step*. So the method of Section 3 has cubic complexity in the sequence length, which makes it challenging to scale. To achieve linear-time inference, we instead approximate by *only marginalizing over k tokens* past the current prefix. We formalize this approximation using the *prefix language* of a grammar.

$$\mathsf{PL}(G) = \{x \mid \exists y \in \Sigma^* : xy \in \mathsf{L}(G)\}$$

In other words, $\mathsf{PL}(G)$ contains all sequences that are the prefix of a sequence in $\mathsf{L}(G)$ (with $\mathsf{L}(G) \subseteq \mathsf{PL}(G)$). We then get the following upper bound.

$$\begin{aligned}
&p(x_{t:n} \in x_{1:t}^{-1}\mathsf{L}(G) \mid x_{1:t}) \\
&\quad = \sum_{x_{t:t+k}} p(x_{t:t+k} \mid x_{1:t})\, p(x_{t+k:n} \in x_{1:t+k}^{-1}\mathsf{L}(G) \mid x_{1:t+k}) \\
&\quad \le \sum_{x_{t:t+k}} p(x_{t:t+k} \mid x_{1:t})\, [\![x_{t:t+k} \in x_{1:t}^{-1}\mathsf{PL}(G)]\!] \\
&\quad = p(x_{t:t+k} \in x_{1:t}^{-1}\mathsf{PL}(G) \mid x_{1:t}) \qquad (2)
\end{aligned}$$

At $k = 0$, our look-ahead window approximation reduces to greedy constrained decoding. Moreover, it monotonically approaches exact HMM marginalization as k increases.

Initially, when the prefix is empty ($x_{1:t} = \epsilon$), we need to compute $p_{\mathsf{HMM}}(x_{1:k+1} \in \mathsf{PL}(G))$. We can apply the method of Section 3 here, by constructing a prefix grammar G' such that $\mathsf{L}(G') = \mathsf{PL}(G)$. There are several constructions to create a prefix grammar; we roughly follow Pasti et al. [2026].

Definition 4 (Prefix grammar)**.** *Given a CBF grammar $G = (\Sigma, \mathcal{N}, S, \mathcal{R})$, we define $G' = (\Sigma, \mathcal{N} \cup \mathcal{N}', \hat{S}, \mathcal{R} \cup \mathcal{R}')$ with $\mathcal{N}' = \{A' \mid A \in \mathcal{N}\}$ the new nonterminals and $\hat{S}$ a new start symbol. $\mathcal{R}'$ contains (1) $A' \to \sigma$ for each terminal rule $A \to \sigma$, (2) $A' \to B'$ for each unary rule $A \to B$, (3) $A' \to B'$ and $A' \to BC'$ for each binary rule $A \to BC$, (4) $\hat{S} \to \epsilon$, and (5) $\hat{S} \to S'$.*

The problem with the above grammar transformation is that it introduces ambiguity: G' can be ambiguous even if G is unambiguous. To avoid this, we assume that G is LL(1).

[1]Position invariance is what gives us quadratic complexity, as opposed to the cubic complexity of the more general construction by Zhang et al. [2025, Theorem 4.2].

Theorem 4. *If G is LL(1), G' is unambiguous.*

Next, we need to handle a non-empty prefix $x_{1:t}$. Because G is LL(1), the parser deterministically consumes $x_{1:t}$ and yields a unique stack $\alpha_1 \cdots \alpha_d \in \mathcal{N}^*$ with $S \overset{*}{\Rightarrow} x_{1:t}\, \alpha_1 \cdots \alpha_d$. The valid continuations of $x_{1:t}$ are exactly the strings derivable from this stack, so we capture them with a fresh nonterminal ξ and the single rule $\xi \to \alpha_1 \cdots \alpha_d$. Applying the prefix-grammar transform of Definition 4 to ξ gives ξ', whose language is precisely the set of non-empty continuations u with $x_{1:t}\, u \in \mathsf{PL}(G)$. By using position invariance to move the window $x_{t:t+k}$ to $x_{1:k+1}$, we get

$$\begin{aligned}
&p(x_{t:t+k} \in x_{1:t}^{-1}\mathsf{PL}(G), z_{t+k} \mid z_t, x_{1:t}) \\
&= p(\xi' \overset{*}{\Rightarrow} x_{1:k+1}, z_{k+1} \mid z_1). \qquad (3)
\end{aligned}$$

See Appendix D for a full derivation. Binarizing ξ and evaluating $p(\xi' \overset{*}{\Rightarrow} x_{1:k+1}, z_{k+1} \mid z_1)$ now proceeds as before. All probabilities for $\alpha_1, \ldots, \alpha_d$ can be precomputed; only the d fresh nonterminals introduced by binarizing ξ remain. The evaluation is therefore a single bottom-up pass over the parser stack (Algorithm 2, Appendix C).

Theorem 5. *Given an HMM with h latent states, an LL(1) grammar, and a window size k, CTRL-CFG samples a length-n sequence in $O(nk^3h^2 + k^2h^3|\mathcal{R}|)$ time complexity and $O(kh^2|\mathcal{N}|)$ space complexity.*

5 EXPERIMENTS

We instantiate CTRL-CFG as a HuggingFace logits processor that re-weights the language model's next-token distribution by the membership upper bound from (2). We evaluate on a grammar-conditioned molecule generation task of Parys et al. [2025]. The goal is to generate a diverse set of valid molecules for three classes (acrylates, chain extenders, and isocyanates). To be valid, a molecule should follow the SMILES grammar, an industry-standard language to write down molecules. Not every sequence generated by the SMILES grammar is a valid molecule, however. We hence use the RDKit library [Landrum et al., 2025] to check ground-truth validity.

Setup. As the language model, we use Gemma 4 E2B-it [Google, 2026]. For each of the 3 molecule classes, we distill an HMM (h=512, 45 EM steps) from 90k Gemma sampled continuations. To evaluate the quality of each method, we sample N=256 molecules per class and report the fraction of *valid* samples, as well as the diversity of the N samples. We consider 3 diversity metrics: the Shannon entropy, the mean pairwise Tanimoto distance over Morgan-3 fingerprints, and the percentage of unique Murcko scaffolds. The last two are chemistry-specific notions of diversity, where the Morgan-3 fingerprints measure more fine-grained and the Murcko scaffolds more structural diversity.

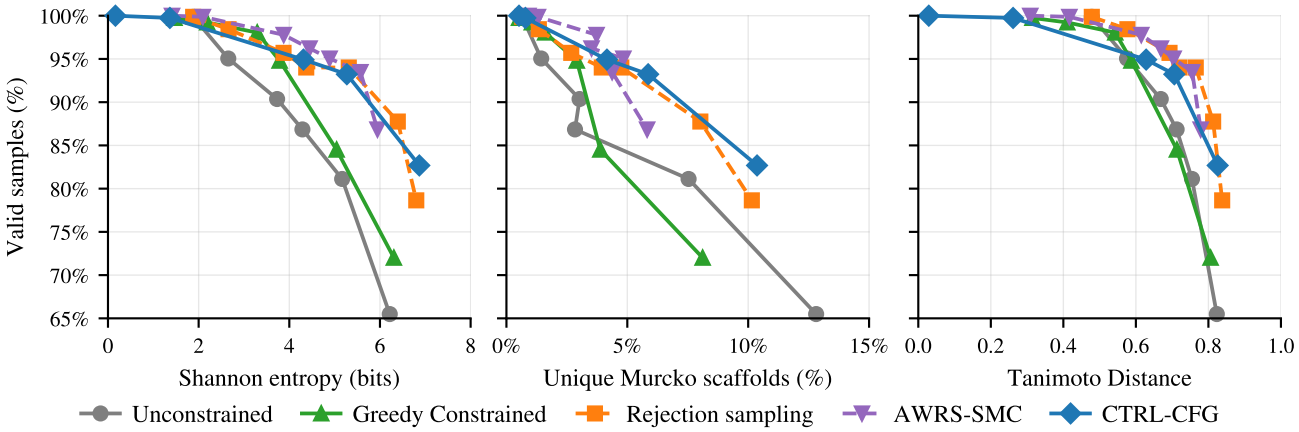

Figure 1: **Validity–diversity trade-off** on the SMILES molecule generation benchmark, swept across different temperatures. Each result is averaged over the three classes (acrylates, chain extenders, and isocyanates). More up and right is better.

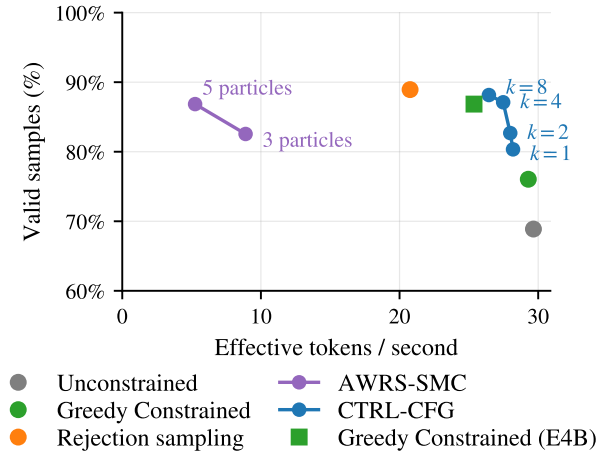

Figure 2: **Validity–throughput trade-off** on the SMILES benchmark at iso-diversity. Validity (% RDKit-parsed) vs. decoding throughput (effective tokens / s) on a single L40S GPU, batch size 1. Temperature for each method is chosen to achieve $H = 6$ bits of entropy. Runtimes are averaged over 90 generations. The CTRL-CFG line sweeps look-ahead depth k from 1 to 8 and for AWRS we report the results for 3 and 5 particles.

Baselines. We compare against four baselines: *unconstrained* sampling; *greedy constrained* sampling (the per-token grammar mask of Dong et al. 2025); *AWRS-SMC*, a state-of-the-art sequential Monte Carlo method [Lipkin et al., 2025]; and *rejection* sampling. As rejection sampling is exact, we can treat it as the ground truth for the best achievable performance with constrained decoding.

CTRL-CFG improves on the validity–diversity trade-off. It is trivial to achieve 100% validity by always returning the same molecule. So Figure 1 sweeps the sampling temperature T from 0.5 to 2.5 for each method to explore the validity–diversity trade-off. CTRL-CFG dominates unconstrained and greedy constrained sampling for each diversity metric. Notably, CTRL-CFG approaches the performance of rejection sampling, indicating that the sliding-window approximation is sufficiently faithful on this benchmark.

CTRL-CFG improves on the validity–speed trade-off. Figure 2 plots validity against decoding throughput at iso-diversity: all methods use a temperature tuned to achieve the same Shannon entropy (6 bits). We report *output* tokens per second, as for example rejection sampling does not use all the generated tokens. Sweeping CTRL-CFG's look-ahead depth from k=1 to k=8 traces a Pareto frontier above and to the right of the baselines.

CTRL-CFG is faster than a larger model with the same quality. As CTRL-CFG incurs overhead for the HMM marginalization, the question arises whether it is not better to instead spend this compute on a larger LLM. In Figure 2, we hence include the Gemma 4 E4B-it model, which has roughly twice as many parameters. On the SMILES benchmark, CTRL-CFG with the E2B-it model and $k = 4$ achieves the same validity and entropy as the larger E4B-it model, while being 8.3% faster.

6 CONCLUSION

We extended constrained autoregressive decoding using HMM-surrogate models from regular languages to context-free grammars. Exact marginalization is cubic in the sequence length, so we proposed a linear-time *look-ahead window* approximation for LL(1) grammars. On grammar-conditioned molecule generation, this dominates greedy per-token grammar masking on each metric and approaches the quality of rejection sampling at much higher throughput.

Acknowledgements

This work was funded in part by the DARPA ANSR and CODORD programs under awards FA8750-23-2-0004 and HR00112590089, and gifts from Cisco Research, Qualcomm, and Amazon. Jaron Maene received funding from the Flemish Government (AI Research Program), the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (Grant agreement No. 101142702). This research was partially conducted while Jaron Maene was visiting UCLA, supported by a travel grant from the Research Foundation Flanders (FWO-Vlaanderen, V410925N).

References

Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. Controllable generation via locally constrained resampling. *Neurosymbolic AI: Foundations and Applications*, pages 159–182, 2026.

Antoine Amarilli, Mikaël Monet, Paul Raphaël, and Sylvain Salvati. On the complexity of language membership for probabilistic words. *CoRR*, abs/2510.08127, 2025. doi: 10.48550/ARXIV.2510.08127.

Annaëlle Baiget, Jaron Maene, Seongmin Lee, Benjie Wang, Guy Van den Broeck, and Miryung Kim. Explainfuzz: Explainable and constraint-conditioned test generation with probabilistic circuits. *arXiv preprint arXiv:2604.06559*, 2026.

Alberto Bertoni, Massimiliano Goldwurm, and Nicoletta Sabadini. The complexity of computing the number of strings of given length in context-free languages. *Theoretical Computer Science*, 86(2):325–342, 1991.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding llms the right way: fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 3658–3673, 2024.

Yixin Dong, Charlie F Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *Proceedings of Machine Learning and Systems*, 7, 2025.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. Jsonschemabench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Google. Gemma 4: Byte for byte, the most capable open models — blog.google. `https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/`, 2026. [Accessed 22-05-2026].

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, 2017.

Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, Riccardo Vianello, sriniker, Peter Gedeck, Gareth Jones, Eisuke Kawashima, NadineSchneider, Dan Nealschneider, Andrew Dalke, tadhurst cdd, Matt Swain, Brian Cole, Samo Turk, Aleksandr Savelev, Alain Vaucher, Maciej Wójcikowski, Ichiru Take, Hussein Faara, Vincent F. Scalfani, Rachel Walker, Daniel Probst, Kazuya Ujihara, Niels Maeder, Jeremy Monat, Juuso Lehtivarjo, and guillaume godin. rdkit/rdkit: 2025_03_5 (q1 2025) release, July 2025. URL `https://doi.org/10.5281/zenodo.16439048`.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023.

Ben 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 *Second Conference on Language Modeling*, 2025.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, et al. Syntactic and semantic control of large language models via sequential monte carlo. In *International Conference on Learning Representations*, volume 2025, pages 64758–64791, 2025.

Reda Marzouk and Colin de La Higuera. Marginal inference queries in hidden markov models under context-free grammar constraints. *arXiv preprint arXiv:2206.12862*, 2022.

Daniel Melcer, Sujan Kumar Gonugondla, Pramuditha Perera, Haifeng Qian, Wen-Hao Chiang, Yanjun Wang, Nihal Jain, Pranav Garg, Xiaofei Ma, and Anoop Deoras. Approximately aligned decoding. *Advances in Neural Information Processing Systems*, 38:11445–11479, 2026.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842, 2019.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion LLMs with context-free grammars. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*, 2025.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. *Advances in Neural Information Processing Systems*, 37:24547–24568, 2024.

Kanghee Park, Timothy Zhou, and Loris D'Antoni. Flexible and efficient grammar-constrained decoding. In *International Conference on Machine Learning*, pages 48262–48275. PMLR, 2025.

PaweĹ Parys, Sairam Vaidya, Taylor Berg-Kirkpatrick, and Loris D'Antoni. Constrained adaptive rejection sampling. *arXiv preprint arXiv:2510.01902*, 2025.

Clemente Pasti, Andreas Opedal, Timothy J. O'Donnell, Ryan Cotterell, and Tim Vieira. Prefix parsing is just parsing. In *Proceedings of ACL*, 2026.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35:9538–9551, 2022.

L Rabiner and B Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

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, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 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/.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.

Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. Tractable control for autoregressive language generation. In *International Conference on Machine Learning*, pages 40932–40945. PMLR, 2023.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. Adaptable logical control for large language models. *Advances in Neural Information Processing Systems*, 37:115563–115587, 2024.

Honghua Zhang, Benjie Wang, Marcelo Arenas, and Guy Van den Broeck. Restructuring tractable probabilistic circuits. In *International Conference on Artificial Intelligence and Statistics*, pages 2566–2574. PMLR, 2025.

Max Zuo, Francisco Piedrahita Velez, Xiaochen Li, Michael Littman, and Stephen Bach. Planetarium: A rigorous benchmark for translating text to structured planning languages. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11223–11240, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.560. URL https://aclanthology.org/2025.naacl-long.560/.

A RELATED WORK

Grammar-constrained decoding. Per-token grammar masking [Scholak et al., 2021, Beurer-Kellner et al., 2024, Park et al., 2025, Dong et al., 2025] is the workhorse of grammar-constrained decoding and is the special case $k{=}0$ of our method. Beam search and lexically constrained decoding [Hokamp and Liu, 2017] pick the most likely sequence under such a mask but inherit its myopia. Unbiased methods rely on variants of rejection sampling [Miao et al., 2019, Park et al., 2024] and Sequential Monte Carlo [Gonzalez et al., 2025, Lipkin et al., 2025] techniques to estimate the membership posterior by sampling. Qin et al. [2022] apply Langevin dynamics, and Ahmed et al. [2026] compile the constraint into a circuit and resample locally. Mündler et al. [2025] adapt rejection sampling to diffusion language models. All of these methods either pay a many-sample cost per step or accept a token-level approximation; our linear-time look-ahead window sits between these regimes.

HMM-based surrogates. Zhang et al. [2023] and Zhang et al. [2024] proposed the use of HMMs as a tractable surrogate for constrained LM decoding, with constraints in conjunctive normal form and as regular expressions, respectively. Closely related, Marzouk and de La Higuera [2022] give an exact quadratic-time algorithm for CFG marginals over HMMs (recapped in Section 3). We build on this for our linear-time look-ahead upper bound. Zhang et al. [2025] give a more general cubic circuit construction for CFGs. Baiget et al. [2026] apply this to learn distributions over grammars for fuzzing. The complexity of probabilistic word problems for context-free grammars is also analyzed by Amarilli et al. [2025]. Our contribution is to turn this line of work into a linear-time, decoder-ready procedure: a look-ahead window approximation that, for LL(1) grammars, compiles to a d-DNNF circuit and is cheap enough to invoke at every generated token.

B GRAMMAR TRANSFORMATIONS

We bring an LL(1) grammar into canonical binary form (Definition 2) with two language-preserving steps. Terminal extraction and binarization preserve LL(1) and add only $O(|\mathcal{R}|)$ symbols and rules. A third and optional pass removes ϵ-productions on the prefix grammar G'.

Terminal extraction. For each terminal σ that occurs in a right-hand side of length at least two, introduce a fresh nonterminal T_σ with the single rule $T_\sigma \to \sigma$ and replace those occurrences of σ by T_σ. Afterwards, terminals appear only in unary rules $A \to \sigma$, as CBF requires. Since T_σ has a unique production with $\mathsf{first}(T_\sigma) = \{\sigma\}$, no parsing decision changes and LL(1) is preserved.

Binarization. We binarize each rule $A \to B_1 B_2 \cdots B_n$ with $n \geq 3$ by replacing it with

$$A \to B_1 C_1, \quad C_1 \to B_2 C_2, \quad \ldots, \quad C_{n-2} \to B_{n-1} B_n,$$

where $C_1, \ldots, C_{n-2}$ are fresh nonterminals. Each C_i carries a single production, so it adds no choice, and $\mathsf{first}(B_1 C_1) = \mathsf{first}(B_1 \cdots B_n)$, so the look-ahead decision at A is unchanged; LL(1) is therefore preserved. A length-n rule yields $n-1$ binary rules, keeping the total size linear.

Epsilon elimination. The complexity argument of Theorem 5 assumes no nullable nonterminals other than (possibly) S. We achieve this with a standard ϵ-elimination pass. Compute the nullable set $\mathsf{Null} := \{A \in \mathcal{N} \mid A \overset{*}{\Rightarrow} \epsilon\}$ as the least fixed point of $A \in \mathsf{Null}$ whenever some rule $A \to \beta$ has every symbol of β in $\mathsf{Null} \cup \{\epsilon\}$. For every binary rule $A \to BC$, add the contractions $A \to C$ (if $B \in \mathsf{Null}$) and $A \to B$ (if $C \in \mathsf{Null}$), and retain the original rule $A \to BC$. Then delete every rule of the form $A \to \epsilon$. Finally, if $S \in \mathsf{Null}$, reinstate $S \to \epsilon$. The result is in CBF, accepts the same language, and at most triples the number of rules.

We do not apply epsilon elimination on the grammar G as it does not preserve LL(1). However, the transformation does preserve unambiguity, meaning it can be applied to G'. To see this, note that in an unambiguous grammar every nullable nonterminal A admits a unique derivation $A \overset{*}{\Rightarrow} \epsilon$. A parse tree in the new grammar then lifts uniquely to one in the original by reinflating each contracted application $A \to B$ or $A \to C$ with the canonical ϵ-derivation of the dropped sibling. Conversely, every old parse tree of a non-empty string collapses to a unique new tree by contracting maximal ϵ-subtrees at each binary rule application. This is a bijection between parse trees of non-empty strings (and the lone parse tree of ϵ is preserved when $S \in \mathsf{Null}$), so the transformed grammar is unambiguous.

C ALGORITHMS

D PROOFS

D.1 DERIVATION FOR EQUATION 3

Below, we give a more detailed derivation of the equality in (3) and explain each step.

$$p(x_{t:t+k} \in x_{1:t}^{-1}\mathsf{PL}(G), z_{t+k} \mid z_t, x_{1:t})$$

The prefix grammar G' of Definition 4 satisfies $\mathsf{L}(G') = \mathsf{PL}(G)$, so prefix membership in G is ordinary membership in G'.

$$= p(x_{t:t+k} \in x_{1:t}^{-1}\mathsf{L}(G'), z_{t+k} \mid z_t, x_{1:t})$$

Algorithm 1 Exact Grammar Marginalization

Require: Unambiguous grammar $G = (\Sigma, \mathcal{N}, S, \mathcal{R})$ in CBF,
Require: HMM with h hidden states,
Require: sequence length $n \geq 1$.
Initialize $\mathcal{W} \in \mathbb{R}^{|\mathcal{N}| \times n \times h \times h}$ to 0.
for $A \in \mathcal{N}$ with $A \overset{*}{\Rightarrow} \epsilon$ **do**
  $\mathcal{W}[A,\ 0] \mathrel{+}= I_{h \times h}$
for $(A \to \sigma) \in \mathcal{R}$ **do**
  $\mathcal{W}[A,\ 1,\ z_1,\ z_2] \mathrel{+}= p(\sigma \mid z_1) \cdot p(z_2 \mid z_1)$
for $\ell \in \{1,\ 2,\ \ldots,\ n-1\}$ **do**
  for $A \in \mathcal{N}$ in bottom-up topological order **do**
    for $(A \to B) \in \mathcal{R}$ with $B \in \mathcal{N}$ **do**
      $\mathcal{W}[A,\ \ell] \mathrel{+}= \mathcal{W}[B,\ \ell]$
    for $(A \to B\, C) \in \mathcal{R}$ **do**
      $\mathcal{W}[A,\ \ell] \mathrel{+}= \sum_{j=0}^{\ell} \mathcal{W}[B,\ j] \cdot \mathcal{W}[C,\ \ell - j]$
return $\mathcal{W}$

Algorithm 2 Stack-based Sliding Window WMC

Require: LL(1) grammar $G = (\Sigma, \mathcal{N}, S, \mathcal{R})$ in CBF,
Require: $\mathcal{W} \in \mathbb{R}^{|\mathcal{N}| \times (k+1) \times h \times h}$ from Algorithm 1 run on the prefix grammar G',
Require: parser state $\alpha_1 \cdots \alpha_d \in \mathcal{N}^*$ from the prefix,
Require: initial forward messages $\alpha \in \mathbb{R}^h$.
Initialize $V \in \mathbb{R}^{(k+1) \times h}$ to 0, and set $V[0] \coloneqq \alpha$.
Initialize $V_\infty \in \mathbb{R}^h$ to 0.
for $i \in \{1, 2, \ldots, d\}$ **do**
  $V_\infty \mathrel{+}= \sum_{j=0}^{k} V[j] \odot \mathcal{W}[\alpha'_i,\ k-j]$
  for $\ell \in \{k,\ k-1,\ \ldots,\ 0\}$ **do**
    $V[\ell] \coloneqq \sum_{j=0}^{\ell} V[j] \cdot \mathcal{W}[\alpha_i,\ \ell - j]$
return $\sum_{i=1}^{h} (V_\infty[i] + \sum_{\ell=0}^{k} V[\ell, i])$

Marginalize over the (unobserved) window tokens $x_{t:t+k}$.

$$= \sum_{x_{t:t+k}} p(x_{t:t+k} \in x_{1:t}^{-1}\mathsf{L}(G'), x_{t:t+k}, z_{t+k} \mid z_t, x_{1:t})$$

The event $\{x_{t:t+k} \in x_{1:t}^{-1}\mathsf{L}(G')\}$ is a deterministic function of $x_{1:t+k}$, so we take it out of the joint as an indicator and apply the definition of the Brzozowski derivative.

$$= \sum_{x_{t:t+k}} p(x_{t:t+k}, z_{t+k} \mid z_t, x_{1:t})\, [\![x_{1:t+k} \in \mathsf{L}(G')]\!]$$

Conditioning on z_t makes the HMM future independent of $x_{1:t}$ (Markov property). We also rewrite membership in $\mathsf{L}(G')$ as derivability from the start symbol S'.

$$= \sum_{x_{t:t+k}} p(x_{t:t+k}, z_{t+k} \mid z_t)\, [\![S' \overset{*}{\Rightarrow} x_{1:t+k}]\!]$$

By Theorem 4 the prefix grammar G' is unambiguous and is parsed deterministically by the LL(1) parser of G, so $S' \overset{*}{\Rightarrow} x_{1:t}\xi' \overset{*}{\Rightarrow} x_{1:t+k}$. The rule $\xi \to \alpha_1 \cdots \alpha_d$ folds the residual parse stack into a single nonterminal.

$$= \sum_{x_{t:t+k}} p(x_{t:t+k}, z_{t+k} \mid z_t)\, [\![x_{1:t}\xi' \overset{*}{\Rightarrow} x_{1:t+k}]\!]$$

The fixed prefix $x_{1:t}$ on both sides of the derivation cancels.

$$= \sum_{x_{t:t+k}} p(x_{t:t+k}, z_{t+k} \mid z_t)\, [\![\xi' \overset{*}{\Rightarrow} x_{t:t+k}]\!]$$

The HMM is time-invariant. So as $x_{1:t}$ does not appear anymore, the joint distribution of $(x_{t:t+k}, z_{t+k})$ given z_t equals the joint distribution of $(x_{1:1+k}, z_{1+k})$ given z_1.

$$= \sum_{x_{1:1+k}} p(x_{1:1+k}, z_{1+k} \mid z_1)\, [\![\xi' \overset{*}{\Rightarrow} x_{1:1+k}]\!]$$

Finally, we collapse the marginal sum.

$$= p(\xi' \overset{*}{\Rightarrow} x_{1:k+1}, z_{k+1} \mid z_1)$$

D.2 PROOF OF THEOREM 4

Proof. We give a proof by contradiction. Assume G' is ambiguous, meaning there is a sequence $x \in \mathsf{L}(G')$ with at least two distinct left-most derivations. We now prove that G is not in LL(1).

Let $\gamma A \mu$ be the point at which the two left-most derivations diverge first (with $\gamma \in \Sigma^*$), where A can be rewritten with two distinct productions $A \to \beta_1$ and $A \to \beta_2$.

$$S' \overset{*}{\Rightarrow} \gamma A \mu \to \gamma \beta_1 \mu \overset{*}{\Rightarrow} \gamma\tau = x$$
$$S' \overset{*}{\Rightarrow} \gamma A \mu \to \gamma \beta_2 \mu \overset{*}{\Rightarrow} \gamma\tau = x$$

We use $x = \gamma\tau$ here with $\tau \in \Sigma^*$, such that $A\mu \overset{*}{\Rightarrow} \tau$. Now it either holds that $A \in \mathcal{N}$ or $A \in \mathcal{N}'$. By construction of G', the primed nonterminal is always last. So it follows that $\mu = \epsilon$ if and only if $A \in \mathcal{N}'$.

Case 1: $A \in \mathcal{N}$. In this case, we have $\mu \neq \epsilon$. So let us call a the first letter in τ. We can now verify the LL(1) properties.

- If $a \in \mathsf{first}(\beta_1) \cap \mathsf{first}(\beta_2)$, the grammar G cannot be in LL(1) due to a first conflict.
- If $a \notin \mathsf{first}(\beta_1)$ and $a \notin \mathsf{first}(\beta_2)$, it follows that $\epsilon \in \mathsf{first}(\beta_1) \cap \mathsf{first}(\beta_2)$, which again is a first conflict.
- If $a \in \mathsf{first}(\beta_1)$ but $a \notin \mathsf{first}(\beta_2)$, β_2 must be nullable and $a \in \mathsf{follow}(\beta_2)$, leading to a follow conflict. The symmetric subcase is analogous.

Case 2: $A \in \mathcal{N}'$. In this case, we have $\mu = \epsilon$, so $\beta_1 \overset{*}{\Rightarrow} \tau$ and $\beta_2 \overset{*}{\Rightarrow} \tau$. By construction of G', the only epsilon-producing rule for a primed nonterminal is $S' \to \epsilon$, so

$\tau \neq \epsilon$ (otherwise $\beta_1 = \beta_2 = \epsilon$, contradicting distinctness). We again call a the first letter in τ, and we know that $a \in \mathsf{first}(\beta_1) \cap \mathsf{first}(\beta_2)$ as nothing follows A. Consider now the corresponding rules in G for $A \to \beta_1$ and $A \to \beta_2$. Let us call these $\hat{A} \to \hat{\beta}_1$ and $\hat{A} \to \hat{\beta}_2$, such that $\hat{A}' = A$.

- In the case where $\hat{A} \to \hat{\beta}_1$ and $\hat{A} \to \hat{\beta}_2$ are different, the same induction as in Lemma 1 lifts $a \in \mathsf{first}(\beta_i)$ to $a \in \mathsf{first}(\hat{\beta}_i)$ for $i \in \{1, 2\}$, giving a first conflict between the two G rules for $\hat{A}$.
- In the other case, $\hat{A} \to BC$ with $\beta_1 = B'$ and $\beta_2 = BC'$ WLOG. From $\beta_2 \overset{*}{\Rightarrow} \tau = uv$ we get $u \in \mathsf{L}_G(B)$ with v a non-empty prefix of $\mathsf{L}_G(C)$, while $\beta_1 \overset{*}{\Rightarrow} \tau$ embeds τ in B's prefix language; hence u is a strict prefix of some string in $\mathsf{L}_G(B)$. The first character of v then lies in $\mathsf{first}(C) \subseteq \mathsf{follow}(B)$ but also extends u inside B, yielding a follow conflict at the nullable position in B responsible for the extension. This contradicts LL(1).

□

Lemma 1. *Given a prefix grammar G' constructed following Definition 4, it holds that* $\mathsf{first}(A') \subseteq \mathsf{first}(A)$ *for any nonterminal* $A \in \mathcal{N}$.

Proof. We apply strong induction on the length of a leftmost derivation $A' \overset{*}{\Rightarrow} a\gamma$, by case on the first rule.

- $A' \to a$ (from $A \to a \in \mathcal{R}$): immediately $A \overset{*}{\Rightarrow} a$.
- $A' \to B'$ (from $A \to B$ or $A \to BC \in \mathcal{R}$): IH on the shorter $B' \overset{*}{\Rightarrow} a\gamma$ gives $B \overset{*}{\Rightarrow} a\beta$, lifted through $A \to B$ or $A \to BC$ to $A \overset{*}{\Rightarrow} a\beta$ or $A \overset{*}{\Rightarrow} a\beta C$.
- $A' \to BC'$ (from $A \to BC \in \mathcal{R}$): leftmost order rewrites the unprimed B first, using only $\mathcal{R}$-rules. Either $B \overset{*}{\Rightarrow} a\beta_B$, giving $A \to BC \overset{*}{\Rightarrow} a\beta_B C$; or $B \overset{*}{\Rightarrow} \epsilon$ and IH on the shorter $C' \overset{*}{\Rightarrow} a\gamma$ gives $C \overset{*}{\Rightarrow} a\beta_C$, giving $A \to BC \overset{*}{\Rightarrow} C \overset{*}{\Rightarrow} a\beta_C$.

Hence $a \in \mathsf{first}(A)$. □

D.3 PROOF SKETCH OF THEOREM 5

The window pre-computation processes each rule for each of k lengths and (for binary rules) over k split points, with HMM states yielding the h^3 factor. The per-step pass touches each of d stack symbols once; for a nonterminal it shifts $V[\ell]$ over $k + 1$ positions through an $h \times h$ matrix, giving $O(k^2h^2)$. The precomputed tensor $\mathcal{W}$ stores one $h \times h$ matrix per (A, ℓ) pair, dominating the memory term at $O(kh^2|\mathcal{N}|)$; the per-step pass adds the parse stack ($O(d)$) and the message vector V ($O(kh)$). Without nullable symbols, every nonterminal on the stack contributes at least one window token, so the relevant prefix of the stack has length at most k. See the epsilon-elimination in Appendix B for how epsilon rules can be eliminated in G' with overhead linear in $|\mathcal{R}|$.