A Tree Sampler for Bounded Context-Free Languages

Breandan Considine
McGill University
bre@ndan.co

Abstract

In the following paper, we present a simple method for sampling trees with or without replacement from BCFLs. A BCFL is a context-free language (CFL) corresponding to an incomplete string with holes, which can be completed by valid terminals. To solve this problem, we introduce an algebraic datatype that compactly represents candidate parse forests for porous strings. Once constructed, sampling trees is a straightforward matter of sampling integers uniformly without replacement, then lazily decoding them into trees.

1 Introduction

A CFG is a quadruple consisting of terminals (Σ), nonterminals (V), productions ($P\colon V \to (V \mid \Sigma)^*$), and a start symbol, ($S$). It is a well-known fact that every CFG is reducible to *Chomsky Normal Form*, $P'\colon V \to (V^2 \mid \Sigma)$, in which every production takes one of two forms, either $w \to xz$, or $w \to t$, where $w, x, z : V$ and $t : \Sigma$. For example, the CFG, $P = \{S \to SS \mid (S) \mid ()\}$, corresponds to the CNF:

$$P' = \left\{ S \to QR \mid SS \mid LR, \quad L \to (, \quad R \to), \quad Q \to LS \right\}$$

Given a CFG, $\mathcal{G}' : \langle \Sigma, V, P, S \rangle$ in CNF, we can construct a recognizer $R : \mathcal{G}' \to \Sigma^n \to \mathbb{B}$ for strings $\sigma : \Sigma^n$ as follows. Let 2^V be our domain, 0 be $\varnothing$, $\oplus$ be $\cup$, and $\otimes$ be defined as:

$$X \otimes Z = \left\{ w \mid \langle x, z \rangle \in X \times Z, (w \to xz) \in P \right\} \tag{1}$$

If we define $\hat{\sigma}_r = \{w \mid (w \to \sigma_r) \in P\}$, then initialize $M^0_{r+1=c}(\mathcal{G}', e) = \hat{\sigma}_r$ and solve for $M = M + M^2$, the fixedpoint M_∞ is fully determined by the superdiagonal entries:

$$M^0 = \begin{pmatrix} \varnothing & \hat{\sigma}_1 & \varnothing & \cdots & \varnothing \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \varnothing \\ & & & \ddots & \hat{\sigma}_n \\ \varnothing & & & & \varnothing \end{pmatrix} \Rightarrow M_\infty = \begin{pmatrix} \varnothing & \hat{\sigma}_1 & \Lambda & \cdots & \Lambda^*_\sigma \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \Lambda \\ & & & \ddots & \hat{\sigma}_n \\ \varnothing & & & & \varnothing \end{pmatrix}$$

Once obtained, the proposition $[S \in \Lambda^*_\sigma]$ decides language membership, i.e., $[\sigma \in \mathcal{L}(\mathcal{G})]$. This procedure is essentially the textbook CYK algorithm in a linear algebraic notation [3]. We are now ready to define the sampling problem as follows:

Definition 1.1 (Completion)**.** Let $\underline{\Sigma} = \Sigma \cup \{\_\}$, where $\_$ represents a hole. We denote $\sqsubseteq\colon \Sigma^n \times \underline{\Sigma}^n$ as the relation $\{\langle \sigma', \sigma \rangle \mid \sigma_i : \Sigma \implies \sigma'_i = \sigma_i\}$ and the set $\{\sigma' : \Sigma \mid \sigma' \sqsubseteq \sigma\}$ as $\mathrm{H}(\sigma)$. Given a porous string $\sigma : \underline{\Sigma}$, we want to sample parse trees generated by $\mathcal{G}$ corresponding to $\sigma' : \mathrm{H}(\sigma) \cap \ell$.

$\mathrm{H}(\sigma) \cap \ell$ is often a large-cardinality set, so we want a procedure which samples trees uniformly without replacement, without enumerating the whole set, parsing and shuffling it.

2 Method

We define an algebraic data type $\mathbb{T}_3 = (V \cup \Sigma) \rightharpoonup \mathbb{T}_2$ where $\mathbb{T}_2 = (V \cup \Sigma) \times (\mathbb{N} \rightharpoonup \mathbb{T}_2 \times \mathbb{T}_2)^*$. Morally, we can think of $\mathbb{T}_2$ as an implicit set of possible trees sharing the same root, and $\mathbb{T}_3$ as a dictionary of possible $\mathbb{T}_2$ values indexed by possible roots, given by a specific CFG under a finite-length porous string. We construct $\hat{\sigma}_r = \Lambda(\sigma_r)$ as follows:

$$\Lambda(s : \underline{\Sigma}) \mapsto \begin{cases} \bigoplus_{s \in \Sigma} \Lambda(s) & \text{if } s \text{ is a hole,} \\ \{\mathbb{T}_2(w, [\langle \mathbb{T}_2(s), \mathbb{T}_2(\varepsilon) \rangle]) \mid (w \to s) \in P\} & \text{otherwise.} \end{cases}$$

This initializes the superdiagonal, enabling us to compute the fixpoint M_∞ by redefining $\oplus, \otimes : \mathbb{T}_3 \times \mathbb{T}_3 \to \mathbb{T}_3$ as:

$$X \oplus Z \mapsto \bigcup_{k \in \pi_1(X \cup Z)} \left\{ k \Rightarrow \mathbb{T}_2(k, x \cup z) \mid x \in \pi_2(X \circ k), z \in \pi_2(Z \circ k) \right\}$$

$$X \otimes Z \mapsto \bigoplus_{(w \to xz) \in P} \left\{ \mathbb{T}_2\big(w, [\langle X \circ x, Z \circ z \rangle]\big) \mid x \in \pi_1(X), z \in \pi_1(Z) \right\}$$

These operators group subtrees by their root nonterminal, then aggregate their children. Instead of tracking sets, each Λ now becomes a dictionary indexed by the root nonterminal, which can be sampled by obtaining $(\Lambda^*_\sigma \circ S) : \mathbb{T}_2$, then recursively choosing twins as we describe in § 2.1, or without replacement via enumeration as described in § 2.2.

2.1 Sampling trees with replacement

Given a probabilistic CFG whose productions indexed by each nonterminal are decorated with a probability vector $\mathbf{p}$ (this may be uniform in the non-probabilistic case), we define a tree sampler $\Gamma : (\mathbb{T}_2 \mid \mathbb{T}_2^2) \rightsquigarrow \mathbb{T}$ which recursively samples children according to a Multinoulli distribution:

$$\Gamma(T) \mapsto \begin{cases} \Gamma\big(\mathrm{Multi}\big(\texttt{children}(T), \mathbf{p}\big)\big) & \text{if } T : \mathbb{T}_2 \\ \big\langle \Gamma\big(\pi_1(T)\big), \Gamma\big(\pi_2(T)\big) \big\rangle & \text{if } T : \mathbb{T}_2 \times \mathbb{T}_2 \end{cases}$$

This is closely related to the generating function for the ordinary Boltzmann sampler from analytic combinatorics,

$$\Gamma C(x) \mapsto \begin{cases} \mathrm{Bern}\left(\frac{A(x)}{A(x)+B(x)}\right) \to \Gamma A(x) \mid \Gamma B(x) & \text{if } \mathcal{C} = \mathcal{A} + \mathcal{B} \\ \big\langle \Gamma A(x), \Gamma B(x) \big\rangle & \text{if } \mathcal{C} = \mathcal{A} \times \mathcal{B} \end{cases}$$

however unlike Duchon et al. [2], our work does not depend on rejection to guarantee exact-size sampling, as all trees contained in $\mathbb{T}_2$ will necessarily be the same width.

*Given a $T : \mathbb{T}_2$, we may also refer to $\pi_1(T), \pi_2(T)$ as $\texttt{root}(T)$ and $\texttt{children}(T)$ respectively, where children are pairs of conjoined twins.

2.2 Sampling without replacement

The type $\mathbb{T}_2$ of all possible trees that can be generated by a CFG in Chomksy Normal Form corresponds to the fixpoints of the following recurrence, which tells us that each $\mathbb{T}_2$ can be a terminal, or a nonterminal and a (possibly empty) sequence of nonterminal pairs and their two children:

$$L(p) = 1 + pL(p) \qquad P(a) = \Sigma + VL\big(V^2P(a)^2\big)$$

Given a $\sigma : \underline{\Sigma}$, we construct $\mathbb{T}_2$ from the bottom-up, and sample from the top-down. Depicted below is a partial $\mathbb{T}_2$, where red nodes are `root`s and blue nodes are `children`:

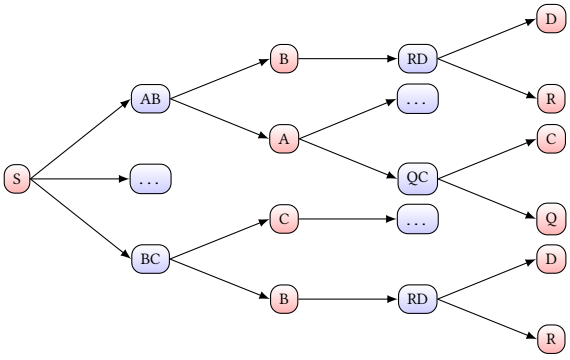

Figure 1. A partial $\mathbb{T}_2$ for the grammar with productions $P = \{S \to BC \mid \ldots \mid AB, B \to RD \mid \ldots, A \to QC \mid \ldots\}$.

To obtain the total number of trees with breadth n, we abstractly parse the porous string using the algebra defined in § 2, and compute the total number of trees using:

$$|T : \mathbb{T}_2| \mapsto \begin{cases} 1 & \text{if } T \text{ is a leaf,} \\ \sum_{\langle T_1, T_2\rangle \in \texttt{children}(T)} |T_1| \cdot |T_2| & \text{otherwise.} \end{cases}$$

To sample all trees in a given $T : \mathbb{T}_2$ uniformly without replacement, we then construct a modular pairing function $\varphi : \mathbb{T}_2 \to \mathbb{Z}_{|T|} \to \texttt{BTree}$, which is defined as follows:

$$\varphi(T : \mathbb{T}_2, i : \mathbb{Z}_{|T|}) \mapsto \begin{cases} \texttt{BTree}\big(\texttt{root}(T)\big) \text{ if } T \text{ is a leaf,} \\ \\ \text{Let } r = |\texttt{children}(T)|, \\ \quad F(n) = \sum_{\langle l,r\rangle \in \texttt{children}[0\ldots n]} |l| \cdot |r|, \\ \quad F^{-1}(u) = \inf\big\{x \mid u \le F(x)\big\}, \\ \quad q = i - F(F^{-1}(i)), \\ \quad l, r = \texttt{children}[t], \\ \quad q_1, q_2 = \big\langle \lfloor \frac{q}{|r|} \rfloor, q \pmod{|r|} \big\rangle, \\ \quad T_1 = \varphi(l, q_1), \\ \quad T_2 = \varphi(r, q_2) \text{ in} \\ \texttt{BTree}\big(\texttt{root}(T), T_1, T_2\big) \text{ otherwise.} \end{cases}$$

Then, instead of sampling trees, we can simply sample integers uniformly without replacement from $\mathbb{Z}_{|T|}$ using the construction defined in § 2, and lazily decode them into trees.

3 Prior work

Our work is closely related to Boltzmann sampling [2] in the case of sampling with replacement, but does not use rejection. Piantodosi [5] defines a similar construction in the case of sampling without replacement, but it assumes the CFG generates an infinite language and its productions have a certain ordering. In the setting where the template contains only holes, BCFL completion coincides with the Chomsky-Schützenberger enumeration theorem [4], which provides a constructive method for counting finite-length words in unambiguous CFLs (i.e., $\ell \cap \Sigma^n$). Our construction is more general, being designed to handle any CFG and template, regardless of ambiguity, finitude, or production ordering.

Loosely adapted from semiring parsing [3] and Valiant's algorithm [6], our parser also supports bounded generation. We construct a nested datatype [1] that compactly represents candidate parse forests and which can be used to sample trees with or without replacement by sampling a finite integer range, enabling communication-free parallelization.

4 Conclusion

We have presented a novel sound and complete algorithm for sampling trees in bounded context-free languages with and without replacement. This technique has applications to code completion and program repair. In future work, we intend to provide a proof of correctness and extend our technique to handle sampling from Boolean and conjunctive languages. A reference implementation for the $\mathbb{T}_2$ datatype is provided in Kotlin and may be found at the URL linked below.[†]

5 Acknowledgements

The author wishes to thank David Bieber for mentioning analytic combinatorics during a sunrise hike and David Yu-Tung Hui for sharing his thoughts on Boltzmann sampling.

References

[1] Richard Bird and Lambert Meertens. 1998. Nested datatypes. In International Conference on Mathematics of Program Construction. Springer, 52–67. http://cs.ox.ac.uk/richard.bird/online/BirdMeertens98Nested.pdf

[2] Philippe Duchon, Philippe Flajolet, et al. 2004. Boltzmann samplers for the random generation of combinatorial structures. Combinatorics, Probability and Computing 13, 4-5 (2004), 577–625.

[3] Joshua Goodman. 1999. Semiring parsing. Computational Linguistics 25, 4 (1999), 573–606. https://aclanthology.org/J99-4004.pdf

[4] Alois Panholzer. 2005. Gröbner Bases and the Defining Polynomial of a CFG Generating Function. J. Autom. Lang. Comb. 10, 1 (2005), 79–97.

[5] Steven T. Piantadosi. 2023. How to enumerate trees from a context-free grammar. arXiv:2305.00522 [cs.CL]

[6] Leslie G Valiant. 1975. General context-free recognition in less than cubic time. Journal of computer and system sciences 10, 2 (1975), 308–315. http://people.csail.mit.edu/virgi/6.s078/papers/valiant.pdf

[†] https://github.com/breandan/galoisenne/blob/3a2e811e8652ba29891aa21789ef0836ed19d257/src/commonMain/kotlin/ai/hypergraph/kaliningraph/parsing/SeqValiant.kt