A Tree Sampler for Bounded Context-Free Languages

Anonymous Author(s)

Abstract

The class of bounded context-free languages (BCFLs) consists of the subset of context-free languages which are finite. We provide a novel algorithm for sampling trees in BCFLs with and without replacement. Once the data structure is constructed, sampling trees is a straightforward matter of sampling integers uniformly without a replacement from a finite range. We demonstrate the utility of this technique on a dataset of Python statements.

1 Introduction

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

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

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

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

If we define $\hat{\sigma}_r := \{w \mid (w \rightarrow \sigma_r) \in P\}$, then initialize $M^0_{r+1=c}(\mathcal{G}', e) := \hat{\sigma}_r$ and solve for the fixpoint $M_\infty = M + M^2$,

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

we obtain the recognizer, $R(\mathcal{G}', \sigma) := S \in \Lambda^*_\sigma? \Leftrightarrow \sigma \in \mathcal{L}(\mathcal{G})?$ Full details of the bisimilarity between parsing and matrix multiplication can be found in Valiant [4] and Lee [1], who shows its time complexity to be $\mathcal{O}(n^\omega)$ where ω is the least matrix multiplication upper bound (currently, $\omega < 2.77$).

2 Method

We define the porous completion problem as follows:

Definition 2.1 (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 $\sigma : \underline{\Sigma}^+$ we want to sample $\sigma' \sim \mathrm{H}(\sigma) \cap \ell$.

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

We define an algebraic data type $\mathbb{T}_3 = (V \cup \Sigma) \rightharpoonup \mathbb{T}_2$ where $\mathbb{T}_2 = (V \cup \Sigma) \times (\mathbb{N} \rightharpoonup \mathbb{T}_2 \times \mathbb{T}_2)$[1]. 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:

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

We then compute the fixpoint M_∞ by redefining $\oplus, \otimes : \mathbb{T}_3 \times \mathbb{T}_3 \rightarrow \mathbb{T}_3$ as follows:

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

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

Decoding trees from $(\Lambda^*_\sigma \circ S) : \mathbb{T}_2$ becomes a straightforward matter of enumeration using a recursive choice function that emits a sequence of binary trees generated by the CFG. We define this construction more precisely in § 2.1.

In our experiments, we provide a comparison of the performance of the SAT algebra and these two semirings, evaluated on a dataset of Python statements.

[1]Hereinafter, given a concrete $T : \mathbb{T}_2$, we shall refer to $\pi_1(T), \pi_2(T)$ as $\texttt{root}(T)$ and $\texttt{children}(T)$ respectively.

2.1 Sampling without replacement

The type $\mathbb{T}_2$ of all possible trees that can be generated by a CFG in Chomksy Normal Form are the fixpoints to the following recurrence:

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

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

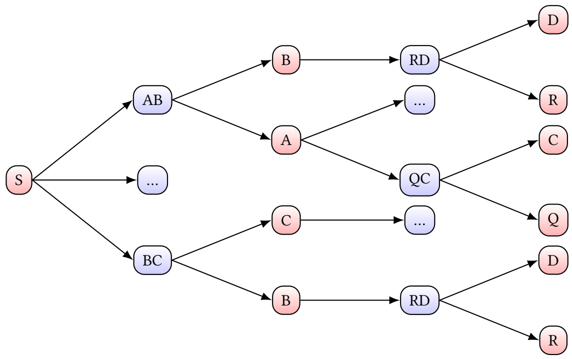

Figure 1. A partial $\mathbb{T}_2$ for the grammar with productions $P = \{S \to BC \mid \ldots \mid AB, B \to RD \mid \ldots, A \to QC \mid \ldots\}$.

The number of binary trees inhabiting a single instance of $\mathbb{T}_2$ is sensitive to the number of nonterminals and rule expansions in the grammar. To obtain the total number of trees with breadth n, we can take the intersection between a CFG and the regular language, $\mathcal{L}(G^\cap) \coloneqq \mathcal{L}(\mathcal{G}) \cap \Sigma^n$, abstractly parse the string containing all holes, let $T = \Lambda^*_{\underline{\sigma}} \circ S$, and compute the total number of trees using the recurrence:

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

To sample all trees in a given $T : \mathbb{T}_2$ uniformly without replacement, we first define a pairing function $\varphi : \mathbb{T}_2 \to \mathbb{Z}_{|T|} \to \texttt{BTree}$ as follows:

$$\varphi(T : \mathbb{T}_2, i : \mathbb{Z}_{|T|}) \mapsto \begin{cases} \Big\langle \texttt{BTree}\big(\texttt{root}(T)\big), i \Big\rangle & \text{if } T \text{ is a leaf,} \\ \text{Let } b = |\texttt{children}(T)|, & \\ \quad q_1, r = \Big\langle \lfloor \frac{i}{b} \rfloor, i \pmod{b} \Big\rangle, & \\ \quad lb, rb = \texttt{children}[r], & \\ \quad T_1, q_2 = \varphi(lb, q_1), & \\ \quad T_2, q_3 = \varphi(rb, q_2) \text{ in} & \\ \Big\langle \texttt{BTree}\big(\texttt{root}(T), T_1, T_2\big), q_3 \Big\rangle & \text{otherwise.} \end{cases}$$

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

Piantodosi define a similar construction, but it assumes the CFL is infinite and makes some additional assumptions about the CFG [3]. We provide a more general construction which works for any CFG. Sampling parse trees in CFGs can be viewed as sampling proofs in a weak kind of proof system [2].

4 Conclusion

We have presented a novel algorithm for sampling trees in bounded context-free languages with and without replacement. This technique has applications to code completion and program repair.

References

[1] Lillian Lee. 2002. Fast context-free grammar parsing requires fast boolean matrix multiplication. Journal of the ACM (JACM) 49, 1 (2002), 1–15. https://arxiv.org/pdf/cs/0112018.pdf

[2] Andreas Opedal, Ran Zmigrod, Tim Vieira, Ryan Cotterell, and Jason Eisner. 2023. Efficient semiringweighted Earley parsing. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), Toronto, Canada.

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

[4] 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