Syntax Error Propagation in Context-Free Languages

Anonymous Author(s)

Abstract

Brzozowski (1964) defines the derivative of a regular language as the suffixes that complete a known prefix. In this work, we establish a Galois connection with Valiant's (1975) fixpoint construction in the context-free setting, and further extend their work into the hierarchy of bounded context-sensitive languages realizable by finite CFL intersection. We show how context-free language recognition can be reduced into a tensor algebra over finite fields, drawing a loose analogy to partial differentiation in Euclidean spaces. In addition to its theoretical contributions, our method has yielded applications to incremental parsing, code completion and program repair. For example, we use it to repair syntax errors and perform sketch-based program synthesis, among other language decision problems.

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 $\sigma_r^{\uparrow} := \{w \mid (w \rightarrow \sigma_r) \in P\}$, then initialize $M^0_{r+1=c}(\mathcal{G}', e) := \sigma_r^{\uparrow}$ and solve for the fixpoint $M^* = M + M^2$,

$$M^0 := \begin{pmatrix} \varnothing & \sigma_1^{\rightarrow} & \varnothing & \cdots & \varnothing \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \varnothing \\ \vdots & & & & \sigma_n^{\uparrow} \\ \varnothing & \cdots & \cdots & \cdots & \varnothing \end{pmatrix} \Rightarrow M^* = \begin{pmatrix} \varnothing & \sigma_1^{\rightarrow} & \Lambda & \cdots & \Lambda_\sigma^* \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \Lambda \\ \vdots & & & & \sigma_n^{\uparrow} \\ \varnothing & \cdots & \cdots & \cdots & \varnothing \end{pmatrix}$$

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 [3], 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

Note that $\bigoplus_{c=1}^{n} M_{r,c} \otimes M_{c,r}$ has cardinality bounded by $|V|$ and is thus representable as a fixed-length vector using the characteristic function, $\mathbb{1}$. In particular, $\oplus, \otimes$ are redefined as $\boxplus, \boxtimes$ over bitvectors so the following diagram commutes,[1]

$$\begin{tikzcd}
\langle \mathcal{G}', \boxed{\Sigma}^{n-1} \rangle \arrow[dr, leftrightarrow] & \text{Set} & \text{Bit} & \langle \mathcal{G}', \Sigma^{n-1} \rangle \arrow[dr] & \text{SAT} \\
\text{Rubix} & M \times M \arrow[r, leftrightarrow, "\mathbb{1}^{2^{n\times n}}", "\mathbb{1}^{-2^{n\times n}}"'] \arrow[d, "\oplus \mid \otimes"] & \mathbb{Z}_2^{|V|^{n\times n}} \times \mathbb{Z}_2^{|V|^{n\times n}} \arrow[rr, tail, "\varphi^{2^{n\times n}}", "\varphi^{-2^{n\times n}}"'] \arrow[d, "\boxed{+} \mid \boxed{*}"] & & \mathcal{M} \times \mathcal{M} \arrow[d, leftrightarrow, "+ \mid *"] \\
\text{Matrix} & 2^V \times 2^V \arrow[r, leftrightarrow, "\mathbb{1}^{2}", "\mathbb{1}^{-2}"'] \arrow[d, "\oplus \mid \otimes"] & \mathbb{Z}_2^{|V|} \times \mathbb{Z}_2^{|V|} \arrow[rr, tail, "\varphi^{2}", "\varphi^{-2}"'] \arrow[d, "\boxed{\boxplus} \mid \boxed{\boxtimes}"] & & \mathcal{V} \times \mathcal{V} \arrow[d, leftrightarrow, "\boxplus \mid \boxtimes"] \\
\text{Vector} & 2^V \arrow[r, leftrightarrow, "\mathbb{1}", "\mathbb{1}^{-1}"'] & \mathbb{Z}_2^{|V|} \arrow[rr, tail, "\varphi", "\varphi^{-1}"'] & & \mathcal{V}
\end{tikzcd}$$

where $\mathcal{V}$ is a function $\mathbb{Z}_2^{|V|} \rightarrow \mathbb{Z}_2$. Note that while always possible to encode $\mathbb{Z}_2^{|V|} \rightarrow \mathcal{V}$ using the identity function, φ^{-1} may not exist, as an arbitrary $\mathcal{V}$ might take on zero, one, or in general, multiple values in $\mathbb{Z}_2^{|V|}$. Although holes may occur anywhere, let us consider two cases in which Σ^+ is strictly left- or right-constrained, i.e., $\boxed{x}\, z, x\, \boxed{z} : \Sigma^{|x|+|z|}$.

Valiant's $\otimes$ operator, which yields the set of productions unifying known factors in a binary CFG, naturally implies the existence of a left- and right-quotient, which yield the set of nonterminals that may appear the right or left side of a known factor and its corresponding root. In other words, a known factor not only implicates subsequent expressions that can be derived from it, but also adjacent factors that may be composed with it to form a given derivation.

Left Quotient	Right Quotient
$\frac{\partial}{\partial \vec{x}} = \left\{ z \mid (w \rightarrow xz) \in P \right\}$	$\frac{\partial}{\partial \vec{z}} = \left\{ x \mid (w \rightarrow xz) \in P \right\}$
$\begin{array}{\|c\|c\|} \hline \boxed{x} & \boxed{w} \\ \hline & z \\ \hline \end{array}$	$\begin{array}{\|c\|c\|} \hline x & \boxed{w} \\ \hline & \boxed{z} \\ \hline \end{array}$

The left quotient coincides with the derivative operator first proposed by Brzozowski [2] and Antimirov [1] over regular languages, lifted into the context-free setting (our work). When the root and LHS are fixed, e.g., $\frac{\partial S}{\partial \vec{x}} : (\overleftarrow{V} \rightarrow S) \rightarrow \vec{\mathcal{V}}$ returns the set of admissible nonterminals to the RHS. One may also consider a gradient operator, $\overleftarrow{\nabla} S : (\overleftarrow{\mathcal{V}} \rightarrow S) \rightarrow \vec{\mathcal{V}}$, which simultaneously tracks the partials with respect to a set of multiple LHS nonterminals produced by a fixed root.

[1] Hereinafter, we use gray highlighting to distinguish between expressions containing only $\boxed{\text{constants}}$ from those which may contain free variables.

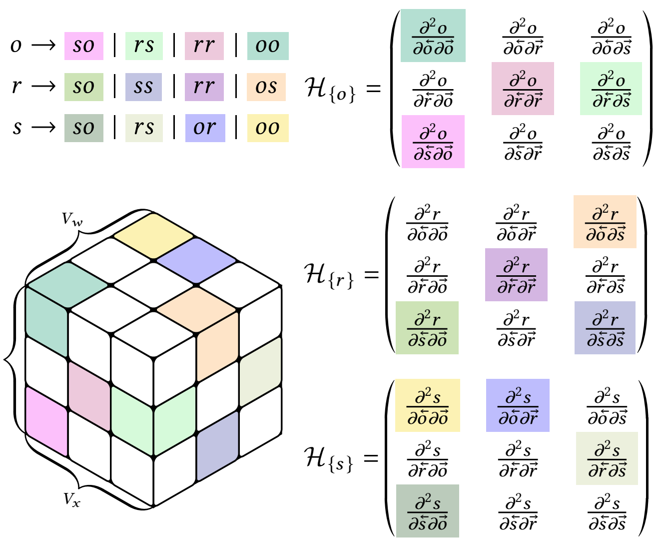

Figure 1. CFGs are witnessed by a rank-3 binary tensor, whose nonzero entries indicate CNF productions. Derivatives in this setting effectively condition the parse tensor. By backpropagating $\mathcal{H}$ across upper-triangular entries of $\mathcal{M}^*$, we constrain the superposition of admissible parse forests.

2.1 Gradient approximation

Given some unparseable string, i.e., $\sigma_1 \ldots \sigma_n : \Sigma^n \cap \mathcal{L}(\mathcal{G})^c$, where should we put holes to obtain a parseable $\sigma' \in \mathcal{L}(\mathcal{G})$? To estimate the effect of perturbing σ on Λ_σ^*, one can either (1) backpropagate ∇S across upper-triangular entries of $\mathcal{M}^*$, or (2) stochastically sample *minibatches* $\boldsymbol{\sigma} : \Sigma^{n \pm k} \sim \Delta_k(\sigma)$ from the Levenshtein k-ball centered on σ, i.e., the space of all edits with Levenshtein distance $\leq k$. Let us consider (2), and let $U : \mathrm{GF}(2^{n \times n})$ be a matrix whose structure is depicted in Eq. 2, where P is a primitive polynomial over $GF(2^n)$ with coefficients $P_{1\ldots n}$ and semiring operators $\oplus := \veebar, \otimes := \wedge$:

$$U^t V = \begin{pmatrix} P_1 & \cdots & \cdots & P_n \\ \top & \circ & \cdots & \circ \\ \circ & \ddots & \ddots & \vdots \\ \circ & \cdots \circ & \top & \circ \end{pmatrix}^t \begin{pmatrix} V_1 \\ \vdots \\ V_n \end{pmatrix} \tag{2}$$

$\mathrm{S} = \begin{pmatrix} V & UV & \cdots & U^{2^n-1}V \end{pmatrix}$ is an ergodic sequence with *full periodicity*, i.e., for all $i, j \in [0, 2^n), \mathrm{S}_i = \mathrm{S}_j \Rightarrow i = j$. To uniformly sample $\boldsymbol{\sigma} \sim \Sigma^d$ without replacement, we form an injection $GF(2^n) \rightharpoonup \Sigma^d$, cycle through S, then reject samples indexing into nonexistent element(s) of Σ. On average, this technique rejects $(1 - |\Sigma| 2^{-\lceil \log_2 |\Sigma| \rceil})^d$ samples and requires $\mathcal{O}(1)$ per sample and $\mathcal{O}(|\Sigma|^n)$ to cover the space. Next, to admit deletion, we augment P with $(\varepsilon^+ \to \varepsilon \mid \varepsilon^+ \varepsilon^+)$ and replace each $(w \to t)$ with $(w \to t\ \varepsilon^+ \mid \varepsilon^+\ t \mid t)$. Finally, to generate $\boldsymbol{\sigma} \sim \Delta_k(\sigma)$, we substitute two adjacent holes $H(\sigma, i) = \sigma_{1\ldots i-1}\ \_\ \_\ \sigma_{i+1\ldots n}$ for each $i \in \binom{n}{1\ldots k}$.

If a solution exists, each edit location has one of six choices:

$$\sigma_1 \ldots \sigma_{i-1}\ \gamma_1 \gamma_2\ \sigma_{i+1} \ldots \sigma_n, \gamma_{1,2} = \varepsilon \tag{3}$$

$$\sigma_1 \ldots \sigma_{i-1}\ \gamma_1 \gamma_2\ \sigma_{i+1} \ldots \sigma_n, \gamma_1 \neq \sigma_i, \gamma_2 = \varepsilon \tag{4}$$

$$\sigma_1 \ldots \sigma_{i-1}\ \gamma_1 \gamma_2\ \sigma_{i+1} \ldots \sigma_n, \gamma_1 = \varepsilon, \gamma_2 \neq \sigma_i \tag{5}$$

$$\sigma_1 \ldots \sigma_{i-1}\ \gamma_1 \gamma_2\ \sigma_{i+1} \ldots \sigma_n, \{\gamma_1, \gamma_2\} \cap \{\varepsilon, \sigma_i\} = \varnothing \tag{6}$$

$$\sigma_1 \ldots \sigma_{i-1}\ \gamma_1 \gamma_2\ \sigma_{i+1} \ldots \sigma_n, \gamma_1 = \sigma_i, \gamma_2 \notin \{\varepsilon, \sigma_i\} \tag{7}$$

$$\sigma_1 \ldots \sigma_{i-1}\ \gamma_1 \gamma_2\ \sigma_{i+1} \ldots \sigma_n, \gamma_1 \notin \{\varepsilon, \sigma_i\}, \gamma_2 = \sigma_i \tag{8}$$

Eq. (3) corresponds to deletion, Eqs. (4, 5, 6) correspond to substitution, and Eqs. (7, 8) correspond to insertion.

2.2 Context-Sensitive Reachability

It is well-known that the family of CFLs is not closed under intersection. For example, consider $\mathcal{L}_\cap := \mathcal{L}(\mathcal{G}_1) \cap \mathcal{L}(\mathcal{G}_1)$:

$$P_1 := \left\{\ S \to LR, \quad L \to ab \mid aLb, \quad R \to c \mid cR\ \right\}$$
$$P_2 := \left\{\ S \to LR, \quad R \to bc \mid bRc, \quad L \to a \mid aL\ \right\}$$

Note that $\mathcal{L}_\cap$ generates the language $\left\{\ a^d b^d c^d \mid d > 0\ \right\}$, which according to the pumping lemma is not context free. We can encode $\bigcap_{i=1}^c \mathcal{L}(\mathcal{G}_i)$ as a polygonal prism with upper-triangular matrices adjoined to each rectangular face. More precisely, we intersect all terminals $\Sigma_\cap := \bigcap_{i=1}^c \Sigma_i$, then for each $t_\cap \in \Sigma_\cap$ and CFG, construct an equivalence class $E(t_\cap, \mathcal{G}_i) = \{w_i \mid (w_i \to t_\cap) \in P_i\}$ and glue them together:

$$\bigwedge_{t \in \Sigma_\cap} \bigwedge_{j=1}^{c-1} \bigwedge_{i=1}^{|\sigma|} E(t_\cap, \mathcal{G}_j) \equiv_{\sigma_i} E(t, \mathcal{G}_{j+1}) \tag{9}$$

Figure 2. Orientations of a $\bigcap_{i=1}^4 \mathcal{L}(\mathcal{G}_i) \cap \Sigma^6$ configuration. As $c \to \infty$, this shape approximates a circular cone whose symmetric axis intersects orthonormal CNF unit productions $w_i \to t_\cap$, with outermost bitvector inhabitants representing $S_i \in \Lambda_\sigma^*$? Equations of this form are equiexpressive with the family of CSLs realizable by finite CFL intersection.

3 Conclusion

Not only is linear algebra over finite fields an expressive language for inference, but also an efficient framework for inference on languages themselves. We illustrate a few of its applications for parsing incomplete strings and repairing syntax errors in context-free and sensitive languages. In contrast with LL and LR-style parsers, our technique can easily recover partial forests from invalid strings by inspecting the structure of M. In future work, we hope to extend our method to more natural grammars like PCFG and LCRS.

References

[1] Valentin Antimirov. 1996. Partial derivatives of regular expressions and finite automaton constructions. Theoretical Computer Science 155, 2 (1996), 291–319.

[2] Janusz A Brzozowski. 1964. Derivatives of regular expressions. Journal of the ACM (JACM) 11, 4 (1964), 481–494. http://maveric.uwaterloo.ca/reports/1964_JACM_Brzozowski.pdf

[3] 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

[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