

# Propagation of Syntax Errors in Context-Sensitive Languages using the Lifted Brzozowski Derivative

Breandan Mark Considine  
McGill University  
bre@ndan.co

Xujie Si  
McGill University  
xsi@cs.mcgill.ca

## Abstract

Brzozowski defines the derivative of a regular language as the suffixes that complete a known prefix. In this work, we establish a Galois connection between Brzozowski’s derivative and Valiant’s 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. By lowering these constructions onto a tensor algebra over finite fields, we draw a loose analogy to partial differentiation in Euclidean spaces. Accompanying its theoretical contributions, our method has also yielded practical 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 common programming-related tasks.

## 1 Introduction

We recall that a CFG is a quadruple consisting of terminals,  $\Sigma$ , nonterminals,  $V$ , productions,  $P : V \rightarrow (V \mid \Sigma)^*$ , and the start symbol,  $S$ . It is a well-known fact that every CFG can be reduced to *Chomsky Normal Form* (CNF),  $P' : V \rightarrow (V^2 \mid \Sigma)$ , in which every production takes one of two forms, either  $w \rightarrow xy$ , or  $w \rightarrow \sigma$ , where  $w, x, y : V$  and  $\sigma : \Sigma$ . For example, the CFG,  $P = \{S \rightarrow SS \mid (S) \mid ()\}$ , corresponds to the CNF:

$$P' = \{S \rightarrow XR \mid SS \mid LR, L \rightarrow (, R \rightarrow ), X \rightarrow LS\}$$

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

$$x \otimes y := \{W \mid \langle X, Y \rangle \in x \times y, (W \rightarrow XY) \in P\} \quad (1)$$

We initialize  $\mathbf{M}_{r,c}^0(\mathcal{G}', \sigma) := \{V \mid c = r + 1, (V \rightarrow \sigma_r) \in P\}$  and search for a matrix  $\mathbf{M}^*$  via fixpoint iteration,

$$\mathbf{M}^* = \begin{pmatrix} \emptyset & \{V\}_{\sigma_1} & \dots & \mathcal{T} \\ \vdots & \vdots & \ddots & \vdots \\ \emptyset & \dots & \{V\}_{\sigma_n} & \emptyset \end{pmatrix} \quad (2)$$

where  $\mathbf{M}^*$  is the least solution to  $\mathbf{M} = \mathbf{M} + \mathbf{M}^2$ . We can then define the recognizer as:  $S \in \mathcal{T} ? \iff \sigma \in \mathcal{L}(\mathcal{G})$ ?

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  $\omega$  is the matrix multiplication bound ( $\omega < 2.77$ ).

Note that  $\bigoplus_{k=1}^n \mathbf{M}_{ik} \otimes \mathbf{M}_{kj}$  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 defined as  $\boxplus, \boxtimes$ , so that the following diagram commutes:

$$\begin{array}{ccc} 2^V \times 2^V & \xrightarrow{\oplus/\otimes} & 2^V \\ \uparrow \mathbb{1}^{-2} \quad \mathbb{1}^2 & & \uparrow \mathbb{1}^{-1} \quad \mathbb{1} \\ \mathbb{B}^{|V|} \times \mathbb{B}^{|V|} & \xrightarrow{\boxplus/\boxtimes} & \mathbb{B}^{|V|} \end{array}$$

This construction can be lifted into the domain of bitvector variables, producing an algebraic expression for each scalar inhabitant of the northeasternmost bitvector, whose solutions correspond to valid parse forests for an incomplete string in the superdiagonal. Consider two cases, where the derivation is left- or right-constrained, i.e.,  $\alpha \gamma, \gamma \alpha$ .<sup>1</sup>

Valiant’s  $\otimes$  operator, which solves for the set of productions unifying known factors in a binary CFG, implies the existence of a left- and right-quotient, which yield the set of nonterminals that may appear to the right- or left-side, respectively, of known factors in a ternary configuration.

Left Quotient

Right Quotient

$$\frac{\partial f}{\partial \bar{x}} = \{y \mid (w \rightarrow xy) \in P\}$$

$$\frac{\partial f}{\partial \bar{y}} = \{x \mid (w \rightarrow xy) \in P\}$$

x	w
	y

x	w
	y

The left quotient coincides with the derivative operator in the context-free setting originally considered by Brzozowski [2] and Antimirov [1] over regular languages.

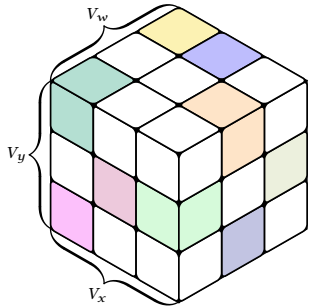
Let  $\mathcal{V}$  represent  $2^V$ . If the root is known, (e.g.,  $S$ ), this represents a scalar-valued function,  $p : \mathcal{V}^* \rightarrow S$ . We may define a gradient operator,  $\nabla : (\mathcal{V} \rightarrow S) \rightarrow \mathcal{V}$  which simultaneously tracks the partials with respect to multiple nonterminals, i.e., ambiguous variables, yielding a known root.

If the root itself is unknown, we can define an operator,  $\mathcal{H} : (\mathcal{V} \rightarrow \mathcal{W}) \rightarrow (\mathcal{V} \times \mathcal{W})$ , which tracks all partial derivatives for a vector variable representing a subset of possible nonterminals. Unlike differential calculus on smooth manifolds, partials in this calculus need not be commutative.

<sup>1</sup>Hereinafter, we shall use gray highlighting to distinguish between bound variables (i.e., constants) and free variables that are unhighlighted.

$$\begin{aligned}
o &\rightarrow \begin{array}{|c|c|c|c|} \hline so & rs & rr & oo \\ \hline \end{array} \\
r &\rightarrow \begin{array}{|c|c|c|c|} \hline so & ss & rr & os \\ \hline \end{array} \\
s &\rightarrow \begin{array}{|c|c|c|c|} \hline so & rs & or & oo \\ \hline \end{array}
\end{aligned}
\quad \mathcal{H}_o = \begin{pmatrix} \frac{\partial^2 o}{\partial \bar{o} \partial \bar{o}} & \frac{\partial^2 o}{\partial \bar{o} \partial \bar{r}} & \frac{\partial^2 o}{\partial \bar{o} \partial \bar{s}} \\ \frac{\partial^2 o}{\partial \bar{r} \partial \bar{o}} & \frac{\partial^2 o}{\partial \bar{r} \partial \bar{r}} & \frac{\partial^2 o}{\partial \bar{r} \partial \bar{s}} \\ \frac{\partial^2 o}{\partial \bar{s} \partial \bar{o}} & \frac{\partial^2 o}{\partial \bar{s} \partial \bar{r}} & \frac{\partial^2 o}{\partial \bar{s} \partial \bar{s}} \end{pmatrix}$$
  

$$\mathcal{H}_r = \begin{pmatrix} \frac{\partial^2 r}{\partial \bar{o} \partial \bar{o}} & \frac{\partial^2 r}{\partial \bar{o} \partial \bar{r}} & \frac{\partial^2 r}{\partial \bar{o} \partial \bar{s}} \\ \frac{\partial^2 r}{\partial \bar{r} \partial \bar{o}} & \frac{\partial^2 r}{\partial \bar{r} \partial \bar{r}} & \frac{\partial^2 r}{\partial \bar{r} \partial \bar{s}} \\ \frac{\partial^2 r}{\partial \bar{s} \partial \bar{o}} & \frac{\partial^2 r}{\partial \bar{s} \partial \bar{r}} & \frac{\partial^2 r}{\partial \bar{s} \partial \bar{s}} \end{pmatrix}$$
  

$$\mathcal{H}_s = \begin{pmatrix} \frac{\partial^2 s}{\partial \bar{o} \partial \bar{o}} & \frac{\partial^2 s}{\partial \bar{o} \partial \bar{r}} & \frac{\partial^2 s}{\partial \bar{o} \partial \bar{s}} \\ \frac{\partial^2 s}{\partial \bar{r} \partial \bar{o}} & \frac{\partial^2 s}{\partial \bar{r} \partial \bar{r}} & \frac{\partial^2 s}{\partial \bar{r} \partial \bar{s}} \\ \frac{\partial^2 s}{\partial \bar{s} \partial \bar{o}} & \frac{\partial^2 s}{\partial \bar{s} \partial \bar{r}} & \frac{\partial^2 s}{\partial \bar{s} \partial \bar{s}} \end{pmatrix}$$


### 1.1 Encoding CFG parsing as SAT solving

By allowing the matrix  $M^*$  in Eq. 2 to contain bitvector variables representing holes in the string and nonterminal sets, we obtain a set of multilinear SAT equations whose solutions exactly correspond to the set of admissible repairs and their corresponding parse forests. Specifically, the repairs coincide with holes in the superdiagonal  $M^*_{r+1=c}$ , and the parse forests occur along the upper-triangular entries  $M^*_{r+1 < c}$ .

$$M^* = \begin{pmatrix} \emptyset & \{V\}_{\sigma_1} & \mathcal{L}_{1,3} & \mathcal{L}_{1,3} & \mathcal{V}_{1,4} & \dots & \mathcal{V}_{1,n} \\ & & \{V\}_{\sigma_2} & \mathcal{L}_{2,3} & & & \\ & & & \{V\}_{\sigma_3} & & & \\ & & & & \mathcal{V}_{4,4} & & \\ & & & & & & \mathcal{V}_{n,n} \\ \emptyset & \dots & \dots & \dots & \dots & \dots & \emptyset \end{pmatrix}$$

Depicted above is a SAT tensor representing  $\sigma_1 \sigma_2 \sigma_3 \dots$  where shaded regions demarcate known bitvector literals  $\mathcal{L}_{r,c}$  (i.e., representing established nonterminal forests) and unshaded regions correspond to bitvector variables  $\mathcal{V}_{r,c}$  (i.e., representing seeded nonterminal forests to be grown). Since  $\mathcal{L}_{r,c}$  are fixed, we precompute them outside the SAT solver.

### 1.2 Deletion, substitution, and insertion

Deletion, substitution and insertion can be simulated by first adding a left- and right-  $\varepsilon$ -production to each unit production:

$$\frac{\Gamma \vdash \varepsilon \in \Sigma}{\Gamma \vdash (\varepsilon^+ \rightarrow \varepsilon \mid \varepsilon^+ \varepsilon^+) \in P} \varepsilon\text{-DUP}$$

$$\frac{\Gamma \vdash (A \rightarrow B) \in P}{\Gamma \vdash (A \rightarrow B \varepsilon^+ \mid \varepsilon^+ B \mid B) \in P} \varepsilon^+\text{-INT}$$

To generate the sketch templates, we substitute two holes at an index-to-be-repaired,  $H(\sigma, i) = \sigma_{1 \dots i-1} \sigma_{i+1 \dots n}$ , then invoke the SAT solver. Five outcomes are then possible:

$$\sigma_1 \dots \sigma_{i-1} \gamma_1 \gamma_2 \sigma_{i+1} \dots \sigma_n, \gamma_{1,2} = \varepsilon \quad (3)$$

$$\sigma_1 \dots \sigma_{i-1} \gamma_1 \gamma_2 \sigma_{i+1} \dots \sigma_n, \gamma_1 \neq \sigma_i, \gamma_2 = \varepsilon \quad (4)$$

$$\sigma_1 \dots \sigma_{i-1} \gamma_1 \gamma_2 \sigma_{i+1} \dots \sigma_n, \gamma_1 = \varepsilon, \gamma_2 \neq \sigma_i \quad (5)$$

$$\sigma_1 \dots \sigma_{i-1} \gamma_1 \gamma_2 \sigma_{i+1} \dots \sigma_n, \gamma_1 = \sigma_i, \gamma_2 \neq \varepsilon \quad (6)$$

$$\sigma_1 \dots \sigma_{i-1} \gamma_1 \gamma_2 \sigma_{i+1} \dots \sigma_n, \gamma_1 \notin \{\varepsilon, \sigma_i\}, \gamma_2 = \sigma_i \quad (7)$$

Eq. (3) corresponds to deletion, Eqs. (4, 5) correspond to substitution, and Eqs. (6, 7) correspond to insertion. This procedure is repeated for all indices in the replacement set. The solutions returned by the SAT solver will be strictly equivalent to handling each edit operation as separate cases.

## 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](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>