

# Backpropagation of Syntax Errors in Context-Free Languages using the Lifted Brzowski Derivative

Breandan Mark Considine  
McGill University  
bre@ndan.co

Jin Guo  
McGill University  
jguo@cs.mcgill.ca

Xujie Si  
McGill University  
xsi@cs.mcgill.ca

## Abstract

Brzowski defines the derivative of a regular language as the suffixes that complete a known prefix. In this work, we establish a Galois connection between parsing with derivatives and Valiant’s fixpoint construction in the context-free setting. By lifting these constructions into the domain of tensors over finite fields, we draw a loose analogy to partial differentiation over Euclidean vector fields. This theory has yielded practical applications for program repair, autocompletion and incremental parsing. For example, we can use it to repair errors, and perform syntax-guided synthesis in context-free languages.

## 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 [?], who shows its time complexity to be  $\mathcal{O}(n^\omega)$  where  $\omega$  is the matrix multiplication bound ( $\omega < 2.77$ ), and Lee [?], who shows that speedups to Boolean matrix multiplication are realizable by CFL parsers.

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:

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

Similarly, this expression can be lifted into the domain of bitvector variables, producing an algebraic expression for the scalar inhabitant representing  $S$  in the northeasternmost bitvector, whose solutions correspond to valid parse forests. We consider two cases, where the string is left- or right-constrained, i.e.,  $\alpha \gamma, \gamma \alpha$  where  $\alpha : \Sigma^k$  and  $\gamma : \Sigma^d$ .<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 a known factor in a binary production.

Left Quotient

Right Quotient

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

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

x	w
	y

x	w
	y

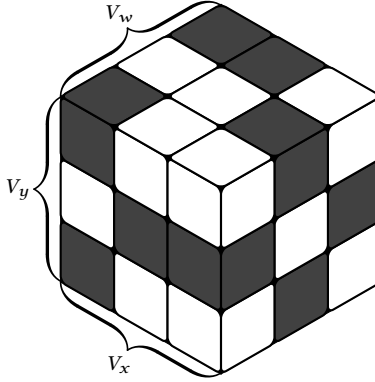
The left quotient coincides with the derivative operator in the context-free setting originally considered by Brzowski [?] and Antimirov [?] over regular languages. These operators in the context-free setting respect linearity. The  $\oplus$  case is trivial. For  $\otimes$ , let us consider the left quotient. Its symmetric case is left as an exercise for the reader:

$$\frac{\partial}{\partial x}(f \otimes g) = \frac{\partial f}{\partial x} \otimes g \oplus f \otimes \frac{\partial g}{\partial x}$$

TODO: prove the product rule holds for CFG reachability.

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

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

$$\begin{array}{lcl}
a & \rightarrow & ac \mid cb \mid bb \mid aa \\
b & \rightarrow & cc \mid ca \mid bb \mid ba \\
c & \rightarrow & cb \mid ac \mid bc \mid aa
\end{array}$$


$$\mathcal{J}_{w_1} = \begin{pmatrix} \frac{\partial a}{\partial a} & \frac{\partial a}{\partial b} & \frac{\partial a}{\partial c} \\ \frac{\partial b}{\partial a} & \frac{\partial b}{\partial b} & \frac{\partial b}{\partial c} \\ \frac{\partial c}{\partial a} & \frac{\partial c}{\partial b} & \frac{\partial c}{\partial c} \end{pmatrix}$$

$$\mathcal{J}_{w_1} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \end{pmatrix}$$

$$\mathcal{J}_{w_2} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \end{pmatrix}$$

$$\mathcal{J}_{w_3} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \end{pmatrix}$$

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, the Jacobian,  $\mathcal{J} : (\mathcal{V} \rightarrow \mathcal{W}) \rightarrow (\mathcal{V} \times \mathcal{W})$ , which tracks all partial derivatives for a vector variable representing a subset of possible nonterminals.

### 1.1 Encoding CFG parsing as SAT solving

By allowing the matrix  $\mathbf{M}^*$  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  $\mathbf{M}_{r+1=c}^*$ , and the parse forests occur along the upper-triangular entries  $\mathbf{M}_{r+1 < c}^*$ .

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

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.