

Syntax Repair as Language Intersection

ANONYMOUS AUTHOR(S)

We introduce a new technique for correcting syntax errors in arbitrary context-free languages. Our work addresses the problem of error correction under a finite number of edits, which we solve by enumerating a language and ranking it by probability. To do this, we adapt the Bar-Hillel construction from formal language theory, guaranteeing the enumerated set is sound and complete with respect to the programming language grammar. This technique also admits a polylogarithmic algorithm for deciding intersection nonemptiness between CFLs and acyclic NFAs, the first of its kind in the parsing literature.

1 INTRODUCTION

During programming, one often encounters scenarios where the editor enters an invalid state. Programmers must spend some time to localize and repair the error before proceeding. We attempt to solve this problem automatically.

2 BACKGROUND

We will first give some background on Brzozowski differentiation. We will use a fragment of the full GRE language with concatenation, conjunction, disjunction.

Definition 2.1 (Generalized Regex). Let E be an expression defined by the grammar:

$$E ::= \emptyset \mid \varepsilon \mid \Sigma \mid E \cdot E \mid E \vee E \mid E \wedge E$$

Semantically, we interpret these expressions as denoting regular languages:

$$\begin{aligned} \mathcal{L}(\emptyset) &= \emptyset & \mathcal{L}(S \cdot T) &= \mathcal{L}(S) \times \mathcal{L}(T)^1 \\ \mathcal{L}(\varepsilon) &= \{\varepsilon\} & \mathcal{L}(S \vee T) &= \mathcal{L}(S) \cup \mathcal{L}(T) \\ \mathcal{L}(a) &= \{a\} & \mathcal{L}(S \wedge T) &= \mathcal{L}(S) \cap \mathcal{L}(T) \end{aligned}$$

Brzozowski introduces the concept of differentiation, which allows us to quotient a regular language by some given prefix.

Definition 2.2 (Brzozowski, 1964). To compute the quotient $\partial_a(L) = \{b \mid ab \in L\}$, we:

$$\begin{aligned} \partial_a(\emptyset) &= \emptyset & \delta(\emptyset) &= \emptyset \\ \partial_a(\varepsilon) &= \emptyset & \delta(\varepsilon) &= \varepsilon \\ \partial_a(b) &= \begin{cases} \varepsilon & \text{if } a = b \\ \emptyset & \text{if } a \neq b \end{cases} & \delta(a) &= \emptyset \\ \partial_a(S \cdot T) &= (\partial_a S) \cdot T \vee \delta(S) \cdot \partial_a T & \delta(S \cdot T) &= \delta(S) \wedge \delta(T) \\ \partial_a(S \vee T) &= \partial_a S \vee \partial_a T & \delta(S \vee T) &= \delta(S) \vee \delta(T) \\ \partial_a(S \wedge T) &= \partial_a S \wedge \partial_a T & \delta(S \wedge T) &= \delta(S) \wedge \delta(T) \end{aligned}$$

Primarily, this gadget was designed to handle membership, for which purpose it has received considerable attention in the parsing literature:

¹Or $\{a \cdot b \mid a \in \mathcal{L}(S) \wedge b \in \mathcal{L}(T)\}$ to be more precise.

THEOREM 2.3 (RECOGNITION). For any regex R and $\sigma : \Sigma^*$, $\sigma \in \mathcal{L}(R) \iff \varepsilon \in \mathcal{L}(\partial_\sigma R)$, where:

$$\partial_\sigma(R) : RE \rightarrow RE = \begin{cases} R & \text{if } \sigma = \varepsilon \\ \partial_b(\partial_a R) & \text{if } \sigma = a \cdot b, a \in \Sigma, b \in \Sigma^* \end{cases}$$

It can also be used, however, to decode witnesses. We will define this process in two steps:

THEOREM 2.4 (GENERATION). For any nonempty (ε, \wedge) -free regex, R , to witness $\sigma \in \mathcal{L}(R)$:

$$\text{follow}(R) : RE \rightarrow 2^\Sigma = \begin{cases} \{R\} & \text{if } R \in \Sigma \\ \text{follow}(S) & \text{if } R = S \cdot T \\ \text{follow}(S) \cup \text{follow}(T) & \text{if } R = S \vee T \end{cases}$$

$$\text{choose}(R) : RE \rightarrow \Sigma^+ = \begin{cases} R & \text{if } R \in \Sigma \\ (s \sim \text{follow}(R)) \cdot \text{choose}(\partial_s R) & \text{if } R = S \cdot T \\ \text{choose}(R' \sim \{S, T\}) & \text{if } R = S \vee T \end{cases}$$

3 LANGUAGE INTERSECTION

THEOREM 3.1 (BAR-HILLEL, 1961). For any context-free grammar (CFG), $G = \langle V, \Sigma, P, S \rangle$, and nondeterministic finite automata, $A = \langle Q, \Sigma, \delta, I, F \rangle$, there exists a CFG $G_\cap = \langle V_\cap, \Sigma_\cap, P_\cap, S_\cap \rangle$ such that $\mathcal{L}(G_\cap) = \mathcal{L}(G) \cap \mathcal{L}(A)$.

Definition 3.2 (Salomaa, 1973). One could construct G_\cap like so,

$$\frac{q \in I \quad r \in F}{(S \rightarrow qSr) \in P_\cap} \sqrt{\quad} \quad \frac{(w \rightarrow a) \in P \quad (q \xrightarrow{a} r) \in \delta}{(qwr \rightarrow a) \in P_\cap} \uparrow \quad \frac{(w \rightarrow xz) \in P \quad p, q, r \in Q}{(pwr \rightarrow (pxq)(qzr)) \in P_\cap} \bowtie$$

However most synthetic productions in P_\cap will be non-generating or unreachable. This naive method will construct a synthetic production for state pairs which are not even reachable, which is clearly excessive.

THEOREM 3.3. For every CFG, G , and every acyclic NFA (ANFA), A , there exists a decision procedure $\varphi : \text{CFG} \rightarrow \text{ANFA} \rightarrow \mathbb{B}$ such that $\varphi(G, A) \models [\mathcal{L}(G) \cap \mathcal{L}(A) \neq \emptyset]$ which requires $\mathcal{O}((\log |Q|)^c)$ time using $\mathcal{O}((|V||Q|)^k)$ parallel processors for some $c, k < \infty$.

PROOF SKETCH. WTS there exists a path $p \rightsquigarrow r$ in A such that $p \in I, r \in F$ where $p \rightsquigarrow r \vdash S$.

There are two cases, at least one of which must hold for $w \in V$ to parse a given $p \rightsquigarrow r$ pair:

- (1) p steps directly to r in which case it suffices to check $\exists a. ((p \xrightarrow{a} r) \in \delta \wedge (w \rightarrow a) \in P)$, or,
- (2) there is some midpoint $q \in Q$, $p \rightsquigarrow q \rightsquigarrow r$ such that $\exists x, z. ((w \rightarrow xz) \in P \wedge \overbrace{p \rightsquigarrow q}^w \rightsquigarrow \overbrace{q \rightsquigarrow r}^z)$.

This decomposition suggests a dynamic programming solution. Let M be a matrix of type $RE^{|Q| \times |Q| \times |V|}$ indexed by Q . Since we assumed δ is acyclic, there exists a topological sort of δ imposing a total order on Q such that M is strictly upper triangular (SUT). Initiate it thusly:

$$M_0[r, c, w] = \bigvee_{a \in \Sigma} \{a \mid (w \rightarrow a) \in P \wedge (q_r \xrightarrow{a} q_c) \in \delta\} \quad (1)$$

The algebraic operations $\oplus, \otimes : RE^{2|V|} \rightarrow RE^{|V|}$ will be defined elementwise:

$$[\ell \oplus r]_w = [\ell_w \vee r_w] \quad (2)$$

$$[\ell \otimes r]_w = \bigvee_{x,z \in V} \{\ell_x \cdot r_z \mid (w \rightarrow xz) \in P\} \quad (3)$$

By slight abuse of notation², we will redefine the matrix exponential over this domain as:

$$\exp(M) = \sum_{i=0}^{\infty} M_0^i = \sum_{i=0}^{|Q|} M_0^i \text{ (since } M \text{ is SUT.)} \quad (4)$$

To solve for the fixpoint, we can instead use exponentiation by squaring:

$$S(2n) = \begin{cases} M_0, & \text{if } n = 1, \\ S(n) + S(n)^2 & \text{otherwise.} \end{cases} \quad (5)$$

Therefor, we only need a maximum of $\lceil \log_2 |Q| \rceil$ sequential steps to reach the fixpoint. Finally,

$$S_{\cap} = \bigvee_{q \in I, q' \in F} \exp(M)[q, q', S] \text{ and } \varphi = [S_{\cap} \neq \emptyset] \quad (6)$$

To decode a witness in case of non-emptiness, we simply choose (S_{\cap}) . \square

4 COMBINATORICS

To enumerate, we first need $|\mathcal{L}(R)|$, which is denoted $|R|$ for brevity.

$$\text{Definition 4.1 (Cardinality). } |R| : RE \rightarrow \mathbb{N} = \begin{cases} 1 & \text{if } R \in \Sigma \\ S \times T & \text{if } R = S \cdot T \\ S + T & \text{if } R = S \vee T \end{cases}$$

THEOREM 4.2 (ENUMERATION). To enumerate, invoke $\bigcup_{i=0}^{|R|} \{enum(R, i)\}$:

$$enum(R, n) : RE \times \mathbb{N} \rightarrow \Sigma^* = \begin{cases} R & \text{if } R \in \Sigma \\ enum(S, \lfloor \frac{n}{|T|} \rfloor) \cdot enum(T, n \bmod |T|) & \text{if } R = S \cdot T \\ enum((S, T)_{\min(1, \lfloor \frac{n}{|S|} \rfloor)}, n - |S| \min(1, \lfloor \frac{n}{|S|} \rfloor)) & \text{if } R = S \vee T \end{cases}$$

²Traditionally, there is a $\frac{1}{k!}$ factor.