Syntax Repair as Language Intersection

ANONYMOUS AUTHOR(S)

We introduce a new technique for correcting syntax errors in arbitrary context-free languages, and by extension, most programming langauges. Our work stems from the observation that syntax errors with a small repair typically have very few unique small repairs, which can usually be enumerated up to a small edit distance then quickly reranked. The enumerated set should contain every repair within a few edits and no invalid repairs. To do so, we adapt the Bar-Hillel construction to acyclic automata and decode it in order of probability. This technique also admits a polylogarithmic decision procedure for finite CFL intersection, the first of its kind.

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 Brozozowski differentiation. We will use a fragment of the full GRE langauge with concatenation, conjunction, disjunction.

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

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

Semantically, we interpret these expressions as denoting regular languages:

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

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:

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

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

[1] Or $\{a \cdot b \mid a \in \mathcal{L}(S) \wedge b \in \mathcal{L}(T)\}$ 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:*

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

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

Theorem 2.4 (Generation). *For any nonempty $(\varepsilon, \wedge)$-free regex, R, to witness $\sigma \in \mathcal{L}(R)$:*

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

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

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,

$$\frac{q \in I \quad r \in F}{(S \to qSr) \in P_\cap}\sqrt{} \qquad \frac{(w \to a) \in P \qquad (q \xrightarrow{a} r) \in \delta}{(qwr \to a) \in P_\cap}\uparrow$$

$$\frac{(w \to xz) \in P \qquad p, q, r \in Q}{(pwr \to (pxq)(qzr)) \in P_\cap}\bowtie$$

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 $\varphi : CFG \to ANFA \to \mathbb{B}$ such that $\varphi(G, A) \models [\mathcal{L}(G) \cap \mathcal{L}(A) \neq \varnothing]$ which requires $\mathcal{O}\big((\log |Q|)^c\big)$ time using $\mathcal{O}\big((|V||Q|)^k\big)$ 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 $\exists a.\big((p \xrightarrow{a} r) \in \delta \wedge (w \to a) \in P\big)$, or,

(2) there is some midpoint $q \in Q$, $p \rightsquigarrow q \rightsquigarrow r$ such that $\exists x, z.\big((w \to xz) \in P \wedge \overbrace{\underbrace{p \rightsquigarrow q}_{x}, \underbrace{q \rightsquigarrow r}_{z}}^{w}\big)$.

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:

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

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

$$[\ell \oplus r]_w = [\ell_w \vee r_w] \tag{2}$$

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

By slight abuse of notation[2], we will redefine the matrix exponential over this domain as:

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

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

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

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

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

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

4 COMBINATORICS

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

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

Theorem 4.2 (Enumeration). *To enumerate, invoke* $\bigcup_{i=0}^{|R|} \{\mathit{enum}(R, i)\}$*:*

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

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