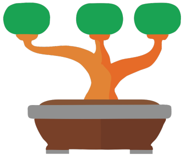
A Tree Sampler for Bounded Context-Free Languages

Breandan Considine

McGill University, Mila IQIA

bre@ndan.co

January 14, 2024



A brief primer on CFL recognition

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

$$s_1 \otimes s_2 \coloneqq \{C \mid \langle A, B\rangle \in s_1 \times s_2, (C \to AB) \in P\}$$

e.g., $\{\text{A} \to \text{BC, C} \to \text{AD, D} \to \text{BA}\} \subseteq \text{P} \vdash \{\text{A, B, C}\} \otimes \{\text{B, C, D}\} = \{\text{A, C}\}$

If we define $\sigma_r^{\nUparrow} \coloneqq \{w \mid (w \to \sigma_r) \in P\}$, then initialize $M^0_{r+1=c}(\mathcal{G}', e) \coloneqq \ \sigma_r^{\nUparrow}$ and solve for the fixpoint $M^* = M + M^2$,

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

$S \Rightarrow^* \sigma \iff \sigma \in \mathcal{L}(\mathcal{G})$ iff $S \in \Lambda_\sigma^*$, i.e., $\mathbb{1}_{\Lambda_\sigma^*}(S) \iff \mathbb{1}_{\mathcal{L}(\mathcal{G})}(\sigma)$.

Satisfiability + holes (our contribution)

- Can be lowered onto a Boolean tensor $\mathbb{B}_2^{n\times n\times|V|}$ (Valiant, 1975)
- Binarized CYK parser can be efficiently compiled to a SAT solver
- Enables sketch-based synthesis in either σ or $\mathcal{G}$: just use variables!
- We simply encode the characteristic function, i.e. $\mathbb{1}_{\subseteq V} : 2^V \to \mathbb{Z}_2^{|V|}$
- $\oplus, \otimes$ are defined as $\boxplus, \boxtimes$, so that the following diagram commutes:

$$\begin{tikzcd}
2^V \times 2^V \arrow[r, "\oplus/\otimes"] \arrow[d, shift right, "\mathbb{1}^{-2}"'] & 2^V \arrow[d, shift right, "\mathbb{1}^{-1}"'] \\
\mathbb{Z}_2^{|V|} \times \mathbb{Z}_2^{|V|} \arrow[r, "\boxplus/\boxtimes"'] \arrow[u, shift right, "\mathbb{1}^2"'] & \mathbb{Z}_2^{|V|} \arrow[u, shift right, "\mathbb{1}"']
\end{tikzcd}$$

- These operators can be lifted into matrices/tensors in the usual way
- In most cases, only a few nonterminals are active at any given time

Satisfiability + holes (our contribution)

Let us consider an example with two holes, $\sigma = 1\ \_\ \_$, and the grammar being $G := \{S \to NON, O \to + \mid \times, N \to 0 \mid 1\}$. This can be rewritten into CNF as $G' := \{S \to NL, N \to 0 \mid 1, O \to \times \mid +, L \to ON\}$. Using the algebra where $\oplus = \cup$, $X \otimes Z = \{\ w \mid \langle x, z\rangle \in X \times Z, (w \to xz) \in P\ \}$, the fixpoint $M' = M + M^2$ can be computed as follows:

	2^V	$\mathbb{Z}_2^{\lvert V\rvert}$	$\mathbb{Z}_2^{\lvert V\rvert} \to \mathbb{Z}_2^{\lvert V\rvert}$
M_0	$\begin{pmatrix} \{N\} & & \\ & \{N,O\} & \\ & & \{N,O\} \end{pmatrix}$	$\begin{pmatrix} \square\blacksquare\square\square & & \\ & \square\blacksquare\blacksquare\square & \\ & & \square\blacksquare\blacksquare\square \end{pmatrix}$	$\begin{pmatrix} V_{0,1} & & \\ & V_{1,2} & \\ & & V_{2,3} \end{pmatrix}$
M_1	$\begin{pmatrix} \{N\} & \varnothing & \\ & \{N,O\} & \{L\} \\ & & \{N,O\} \end{pmatrix}$	$\begin{pmatrix} \square\blacksquare\square\square & \square\square\square\square & \\ & \square\blacksquare\blacksquare\square & \blacksquare\square\square\square \\ & & \square\blacksquare\blacksquare\square \end{pmatrix}$	$\begin{pmatrix} V_{0,1} & V_{0,2} & \\ & V_{1,2} & V_{1,3} \\ & & V_{2,3} \end{pmatrix}$
M_∞	$\begin{pmatrix} \{N\} & \varnothing & \{S\} \\ & \{N,O\} & \{L\} \\ & & \{N,O\} \end{pmatrix}$	$\begin{pmatrix} \square\blacksquare\square\square & \square\square\square\square & \square\square\square\blacksquare \\ & \square\blacksquare\blacksquare\square & \blacksquare\square\square\square \\ & & \square\blacksquare\blacksquare\square \end{pmatrix}$	$\begin{pmatrix} V_{0,1} & V_{0,2} & V_{0,3} \\ & V_{1,2} & V_{1,3} \\ & & V_{2,3} \end{pmatrix}$

Semiring algebras: Part I

The prior solution tell us whether $A(\sigma)$ is nonempty, but forgets the solution(s). To solve for $A(\sigma)$, a naïve approach accumulates a mapping of nonterminals to sets of strings. Letting $D = V \to \mathcal{P}(\Sigma^*)$, we define $\oplus, \otimes : D \times D \to D$. Initially, we construct $M_0[r+1=c] = p(\sigma_r)$ using:

$$p(s:\Sigma) \mapsto \{w \mid (w \to s) \in P\} \text{ and } p(\_) \mapsto \bigcup_{s\in\Sigma} p(s)$$

$p(\cdot)$ constructs the superdiagonal, then we solve for Λ_σ^* using the algebra:

$$X \oplus Z \mapsto \left\{w \overset{+}{\Rightarrow} (X \circ w) \cup (Z \circ w) \mid w \in \pi_1(X \cup Z)\right\}$$

$$X \otimes Z \mapsto \bigoplus_{w,x,z} \left\{w \overset{+}{\Rightarrow} (X \circ x)(Z \circ z) \mid (w \to xz) \in P, x \in X, z \in Z\right\}$$

After M_∞ is attained, the solutions can be read off via $\Lambda_\sigma^* \circ S$. The issue here is exponential growth when eagerly computing the transitive closure.

Semiring algebras: Part II

The prior encoding can be improved using an ADT $\mathbb{T}_3 = (V \cup \Sigma) \rightharpoonup \mathbb{T}_2$ where $\mathbb{T}_2 = (V \cup \Sigma) \times (\mathbb{N} \rightharpoonup \mathbb{T}_2 \times \mathbb{T}_2)$. We construct $\hat{\sigma}_r = \dot{p}(\sigma_r)$ using:

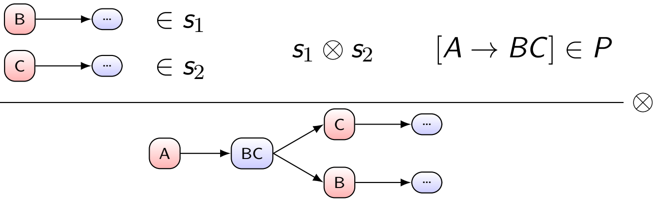
$$\dot{p}(s : \Sigma) \mapsto \Big\{\mathbb{T}_2\big(w, \big[\langle \mathbb{T}_2(s), \mathbb{T}_2(\varepsilon)\rangle\big]\big) \mid (w \to s) \in P\Big\} \text{ and } \dot{p}(\_) \mapsto \bigoplus_{s \in \Sigma} p(s)$$

We then compute the fixpoint M_∞ by redefining $\oplus, \otimes : \mathbb{T}_3 \times \mathbb{T}_3 \to \mathbb{T}_3$ as:

$$X \oplus Z \mapsto \bigcup_{k \in \pi_1(X \cup Z)} \Big\{k \Rightarrow \mathbb{T}_2(k, x \cup z) \mid x \in \pi_2(X \circ k), z \in \pi_2(Z \circ k)\Big\}$$

$$X \otimes Z \mapsto \bigoplus_{(w \to xz) \in P} \Big\{\mathbb{T}_2\big(w, \big[\langle X \circ x, Z \circ z\rangle\big]\big) \mid x \in \pi_1(X), z \in \pi_1(Z)\Big\}$$

$\mathbb{T}_3$ join and merge semantics

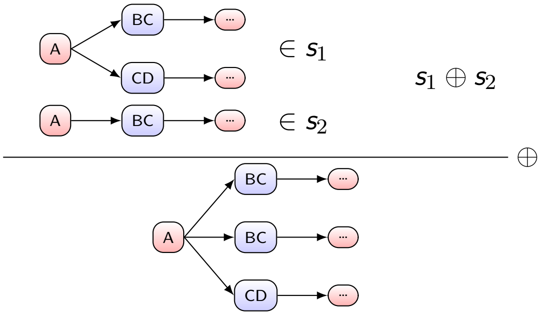


$\in s_1$

$\in s_2$

$s_1 \otimes s_2 \qquad [A \to BC] \in P$

$\otimes$



$\in s_1$

$\in s_2$

$s_1 \oplus s_2$

$\oplus$

Semiring algebras: Part III

$L(p) = 1 + pL(p) \qquad P(a) = \Sigma + V + VL\big(P(a)^2\big)$

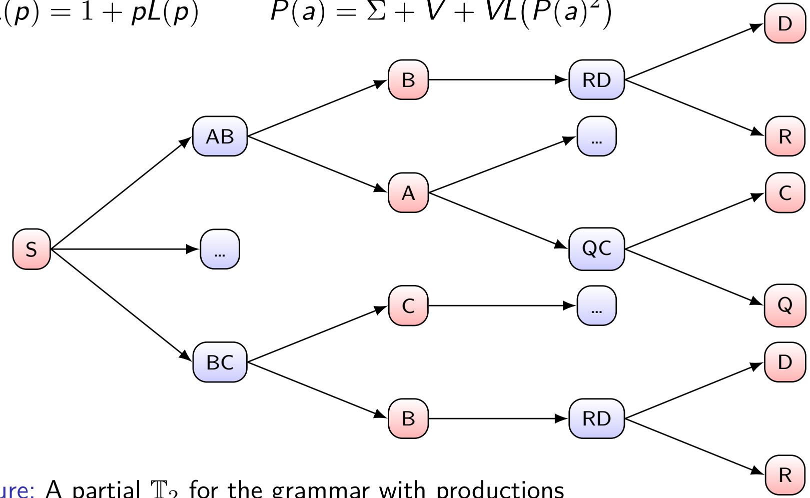


Figure: A partial $\mathbb{T}_2$ for the grammar with productions $P = \{S \to BC \mid \ldots \mid AB, B \to RD \mid \ldots, A \to QC \mid \ldots\}$.

Sampling trees with replacement

Given a probabilistic CFG whose productions indexed by each nonterminal are decorated with a probability vector $\mathbf{p}$ (this may be uniform in the non-probabilistic case), we define a tree sampler $\Gamma : \mathbb{T}_2 \rightsquigarrow \mathbb{T}$ which recursively samples children according to a Multinoulli distribution:

$$\Gamma(T) \mapsto \begin{cases} \text{Multi}\big(\texttt{children}(T), \mathbf{p}\big) & \text{if } T \text{ is a } \texttt{root} \\ \big\langle \Gamma\big(\pi_1(T)\big), \Gamma\big(\pi_2(T)\big)\big\rangle & \text{if } T \text{ is a } \texttt{child} \end{cases}$$

This is closely related to the generating function for the ordinary Boltzmann sampler from analytic combinatorics,

$$\Gamma C(x) \mapsto \begin{cases} \text{Bern}\left(\frac{A(x)}{A(x)+B(x)}\right) \to \Gamma A(x) \mid \Gamma B(x) & \text{if } \mathcal{C} = \mathcal{A} + \mathcal{B} \\ \big\langle \Gamma A(x), \Gamma B(x) \big\rangle & \text{if } \mathcal{C} = \mathcal{A} \times \mathcal{B} \end{cases}$$

however unlike Duchon et al. (2004), rejection is unnecessary to ensure exact-size sampling, as all trees in $\mathbb{T}_2$ will necessarily be the same size.

A pairing function for replacement-free tree sampling

The total number of trees induced by a given sketch template is given by:

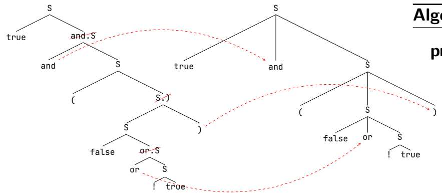
$$|T : \mathbb{T}_2| \mapsto \begin{cases} 1 & \text{if } T \text{ is a leaf,} \\ \sum_{\langle T_1, T_2 \rangle \in \texttt{children}(T)} |T_1| \cdot |T_2| & \text{otherwise.} \end{cases}$$

To sample from $\mathbb{T}_2$ without replacement, we define a pairing function:

$$\varphi^{-1}(T : \mathbb{T}_2, i : \mathbb{Z}_{|T|}) \mapsto \begin{cases} \Big\langle \texttt{BTree}\big(\texttt{root}(T)\big), i \Big\rangle & \text{if } T \text{ is a leaf,} \\ \text{Let } b = |\texttt{children}(T)|, & \\ \quad q_1, r = \big\langle \lfloor \frac{i}{b} \rfloor, i \pmod{b} \big\rangle, & \\ \quad lb, rb = \texttt{children}[r], & \\ \quad T_1, q_2 = \varphi^{-1}(lb, q_1), & \\ \quad T_2, q_3 = \varphi^{-1}(rb, q_2) \text{ in} & \\ \Big\langle \texttt{BTree}\big(\texttt{root}(T), T_1, T_2\big), q_3 \Big\rangle & \text{otherwise.} \end{cases}$$

Chomsky Denormalization

Chomksy normalization is needed for matrix-based parsing, however produces lopsided parse trees. We can denormalize them using a simple recusive procedure to restore the natural shape of the original CFG:



Pre-Denormalization Post-Denormalization

Algorithm Rewrite procedure for tree denormalization

```
procedure $\textsc{Cut}$(t: Tree)
    stems $\leftarrow \{\, \textsc{Cut}(\mathtt{c}) \mid \mathtt{c} \in$ t.children $\}$
    if t.root $\in (V_{\mathcal{G}'} \setminus V_{\mathcal{G}})$ then
        return stems
    else
        return $\{$ Tree(t.root, stems) $\}$
    end if
end procedure
```

All synthetic nonterminals are excised during Chomsky denormalization. Rewriting improves legibility but does not alter the underlying semantics.

Error Correction: Sketch Templates

To sample $\boldsymbol{\sigma} \sim \Delta_q(\underset{\sim}{\sigma})$, we could enumerate a series of sketch templates $H(\sigma, i) = \sigma_{1\ldots i-1} \text{\_ \_} \sigma_{i+1\ldots n}$ for each $i \in \cdot \in \left\{ {n \atop d} \right\}$ and $d \in 1 \ldots q$, then solve for $\mathcal{M}^*_{\boldsymbol{\sigma}}$. If $S \in \Lambda^*_{\boldsymbol{\sigma}}?$ has a solution, each edit in each $\sigma' \in \boldsymbol{\sigma}$ will match exactly one of the following seven edit patterns:

$$\text{Deletion} = \left\{ \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots \quad \gamma_{1,2} = \varepsilon \right.$$

$$\text{Substitution} = \begin{cases} \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots & \gamma_1 \neq \varepsilon \wedge \gamma_2 = \varepsilon \\ \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots & \gamma_1 = \varepsilon \wedge \gamma_2 \neq \varepsilon \\ \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots & \{\gamma_1, \gamma_2\} \cap \{\varepsilon, \sigma_i\} = \varnothing \end{cases}$$

$$\text{Insertion} = \begin{cases} \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots & \gamma_1 = \sigma_i \wedge \gamma_2 \notin \{\varepsilon, \sigma_i\} \\ \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots & \gamma_1 \notin \{\varepsilon, \sigma_i\} \wedge \gamma_2 = \sigma_i \\ \ldots \sigma_{i-1}\, \gamma_1\, \gamma_2\, \sigma_{i+1} \ldots & \gamma_{1,2} = \sigma_i \end{cases}$$

But this is very expensive, requiring $\widetilde{\mathcal{O}}\left(\binom{n}{d} |\Sigma + 1|^{2d}\right)$ to search $\left\{ {n \atop d} \right\} \times \Sigma_\varepsilon^{2d}$.

Error Correction: d-Subset Sampling

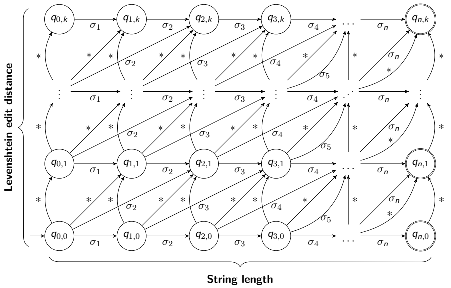
Next, suppose $U : \mathbb{Z}_2^{m\times m}$ is a matrix whose structure is shown in Eq. 1, wherein C is a primitive polynomial over $\mathbb{Z}_2^m$ with coefficients $C_{1\ldots m}$ and semiring operators $\oplus := \veebar, \otimes := \land$:

$$U^t V = \begin{pmatrix} C_1 & \cdots & \cdots & \cdots & C_m \\ \top & \circ & \cdots & \cdots & \circ \\ \circ & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \circ & \cdots & \circ & \top & \circ \end{pmatrix}^t \begin{pmatrix} V_1 \\ \vdots \\ \vdots \\ \vdots \\ V_m \end{pmatrix} \tag{1}$$

Since C is primitive, the sequence $\mathbf{S} = (U^{0\ldots 2^m-1}V)$ must have *full periodicity*, i.e., for all $i, j \in [0, 2^m)$, $\mathbf{S}_i = \mathbf{S}_j \Rightarrow i = j$. To uniformly sample $\boldsymbol{\sigma}$ without replacement, we first form an injection $\mathbb{Z}_2^m \rightharpoonup \left\{ {n \atop d} \right\} \times \Sigma_\varepsilon^{2d}$ using a combinatorial number system, cycle over $\mathbf{S}$, then discard samples which have no witness in $\left\{ {n \atop d} \right\} \times \Sigma_\varepsilon^{2d}$. This method requires $\widetilde{\mathcal{O}}(1)$ per sample and $\widetilde{\mathcal{O}}\left(\binom{n}{d}|\Sigma + 1|^{2d}\right)$ to exhaustively search $\left\{ {n \atop d} \right\} \times \Sigma_\varepsilon^{2d}$.

Levenshtein reachability and monotone infinite automata

MAIA



CFG

$$S \Rightarrow \{\cdot \in Q \mid \delta(\cdot, q_{n,0}) \leq k\}$$
$$* \Rightarrow \{\cdot \in \Sigma\}$$
$$\big\{q_{i,j} \Rightarrow \{q_{i,j-1}*\} \mid i,j \in [1,n] \times [1,k]\big\}$$
$$\big\{q_{i,j} \Rightarrow \{q_{i-1,j-1}*\} \mid i,j \in [1,n] \times [1,k]\big\}$$
$$\big\{q_{i,j} \Rightarrow \{q_{i-1,j}\sigma_i\} \mid i,j \in [1,n] \times [0,k]\big\}$$
$$\big\{q_{i,j} \Rightarrow \{q_{i-2,j-1}\sigma_i\} \mid i,j \in [2,n] \times [1,k]\big\}$$

Figure: Bounded Levenshtein reachability from $\sigma : \Sigma^n$ is expressible as either a monotone acyclic infinite automata (MAIA) populated by accept states within radius k of $S = q_{n,0}$ (left), or equivalently, a left-linear CFG whose productions bisimulate the transition dynamics up to a fixed horizon (right), accepting only strings within Levenshtein radius k of σ.

The Chomsky-Levenshtein-Bar-Hillel Construction

The original Bar-Hillel construction provides a way to construct a grammar for the intersection of a regular and context-free langauge.

$$\frac{q \in I \quad r \in F}{\big(S \to qSr\big) \in P_\cap} \qquad \frac{(q \overset{a}{\to} r) \in \delta}{\big(qar \to a\big) \in P_\cap} \qquad \frac{(w \to xz) \in P \qquad p, q, r \in Q}{\big(pwr \to (pxq)(qzr)\big) \in P_\cap}$$

The Levenshtein automata is another kind of lattice, not in the order-theoretic sense, but in the automata-theoretic sense.

$$\frac{s \in \Sigma \quad i \in [0, n] \quad j \in [1, k]}{(q_{i,j-1} \overset{s}{\to} q_{i,j}) \in \delta} \uparrow \qquad \frac{s \in \Sigma \quad i \in [1, n] \quad j \in [1, k]}{(q_{i-1,j-1} \overset{s}{\to} q_{i,j}) \in \delta} \nearrow$$

$$\frac{s = \sigma_i \quad i \in [1, n] \quad j \in [0, k]}{(q_{i-1,j} \overset{s}{\to} q_{i,j}) \in \delta} \to \qquad \frac{s = \sigma_i \quad i \in [2, n] \quad j \in [1, k]}{(q_{i-2,j-1} \overset{s}{\to} q_{i,j}) \in \delta} \nearrow$$

$$\frac{}{q_{0,0} \in I}\ \textsc{Init} \qquad \frac{q_{i,j} \qquad |n - i + j| \leq k}{q_{i,j} \in F}\ \textsc{Done}$$

Abbreviated history of algebraic parsing

- Chomsky & Schützenberger (1959) - The algebraic theory of CFLs
- Cocke–Younger–Kasami (1961) - Bottom-up matrix-based parsing
- Brzozowski (1964) - Derivatives of regular expressions
- Earley (1968) - top-down dynamic programming (no CNF needed)
- Valiant (1975) - first realizes the Boolean matrix correspondence
  - Naïvely, has complexity $\mathcal{O}(n^4)$, can be reduced to $\mathcal{O}(n^\omega)$, $\omega < 2.763$
- Lee (1997) - Fast CFG Parsing $\Longleftrightarrow$ Fast BMM, formalizes reduction
- Might et al. (2011) - Parsing with derivatives (Brzozowski $\Rightarrow$ CFL)
- Bakinova, Okhotin et al. (2010) - Formal languages over GF(2)
- Bernady & Jansson (2015) - Certifies Valiant (1975) in Agda
- Cohen & Gildea (2016) - Generalizes Valiant (1975) to parse and recognize mildly context sensitive languages, e.g. LCFRS, TAG, CCG
- Considine, Guo & Si (2022) - SAT + Valiant (1975) + holes
- **Considine (2023)** - $\mathbb{T}_3$ **completion + distinct tree sampling**

Uniform sampling benchmark on natural syntax errors

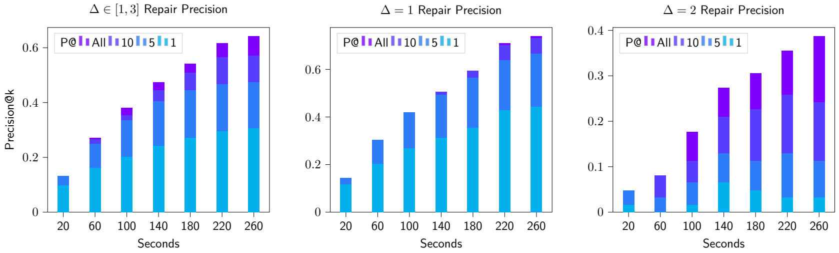


Figure: Repairs discovered before the latency cutoff are reranked based on their tokenwise perplexity and compared for an exact lexical match with the human repair at or below rank k. We note that the uniform sampling procedure is not intended to be used in practice, but provides a baseline for the empirical density of the admissible set, and an upper bound on the latency required to attain a given precision.

Adaptive sampling benchmark on natural syntax errors

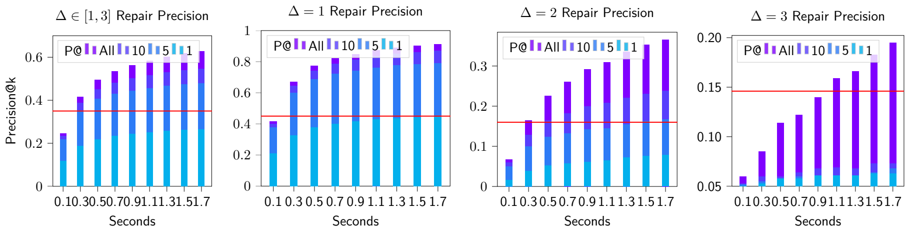


Figure: Adaptive sampling repairs. The red line indicates Seq2Parse precision@1 on the same dataset. Since it only supports generating one repair, we do not report precision@k or the intermediate latency cutoffs.

Precision on Error Correction in Synthetic Language

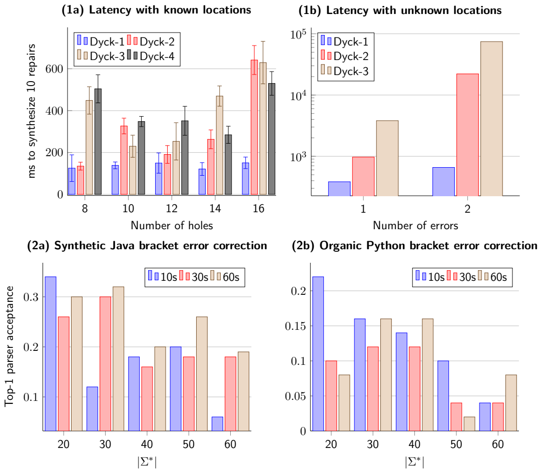


Figure: Benchmarking bracket correction latency and accuracy across two bracketing langauges, one generated from Dyck-n, and the second uses an abstracted source code snippet with imbalanced parentheses.

Uniform sampling benchmark

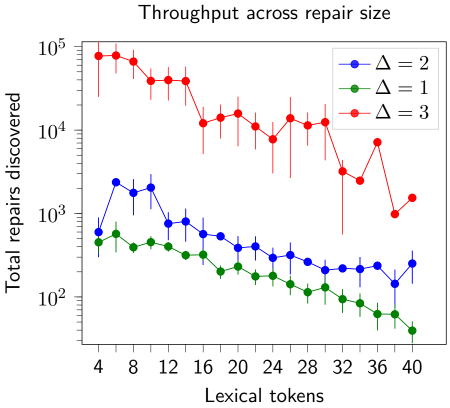
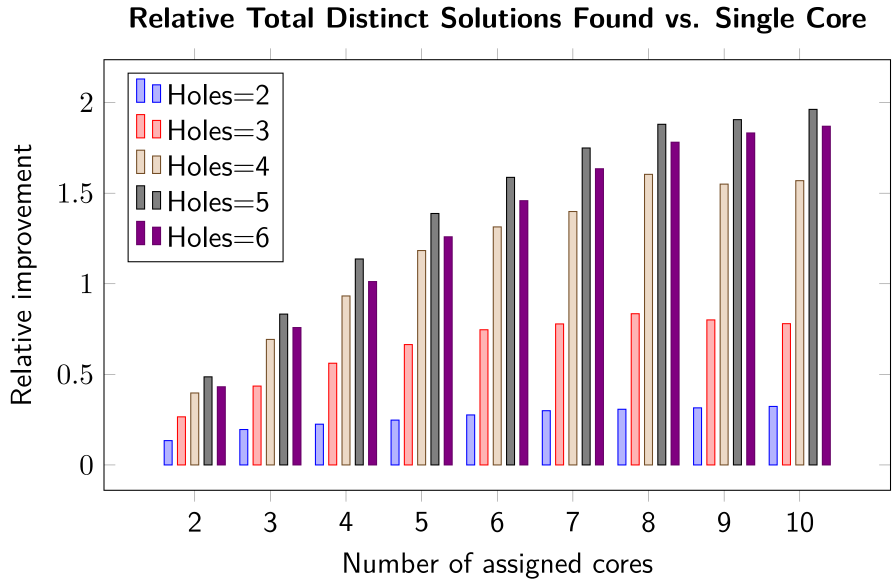


Figure: We evaluate throughput by sampling edits across invalid strings $|\underset{\sim}{\sigma}| \leq 40$ from the StackOverflow dataset of varying length, and measure the total number of syntactically valid edits discovered, as a function of string length and language edit distance $\Delta \in [1, 3]$. Each trial is terminated after 10 seconds, and the experiment is repeated across 7.3k total repairs.

Multicore Scaling Results (aarch64)



Special thanks

David Bieber, David Yu-Tung Hui
Jin Guo, Xujie Si

McGill UNIVERSITY Mila

Learn more at:

https://tidyparse.github.io