Tidyparse: Real-Time Context-free Error Correction and the Bounded Levenshtein Reachability Problem

Breandan Considine, Jin Guo, Xujie Si

McGill University, Mila IQIA

bre@ndan.co

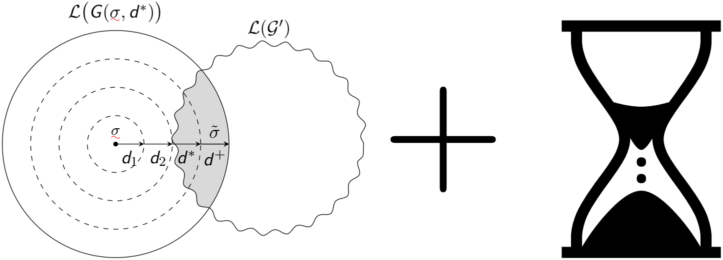
October 19, 2023

Overview

Our Contribution

Helps novice programmers fix syntax errors in source code. We do so by solving the **Realtime Bounded Levenshtein Reachability Problem**:

Given a linear conjunctive language $\ell : \mathcal{L}(\mathcal{G})$ and an invalid string $\underset{\sim}{\sigma} : \Sigma^*$, find every syntactically admissible edit $\tilde{\sigma}$ satisfying $\{\tilde{\sigma} \in \ell \mid \Delta(\underset{\sim}{\sigma}, \ell) < r\}$, ranked by a probability metric Δ, under hard realtime constraints.



Natural language: *Rapidly finds syntactically valid edits within a small neighborhood, ranked by tokenwise similarity and statistical likliehood.*

On the virtues of pragmatism

pragmatism: *a reasonable and logical way of doing things or of thinking about problems that is based on dealing with specific situations instead of on ideas and theories.*

- Often framed as a compromise or concession, e.g., *"It's not ideal, but let's be pragmatic."*
- Taken seriously, pragmatism is difficult because it requires accurately modeling and weighing the needs of multiple stakeholders.
- Pragmatism is a principled approach to problem solving.
- Putting it into practice requires knowing your customer, understanding their workflow, considering the most appropriate solution out of many possible alternatives.

Syntax repair as a Gricean language game

- Imagine a game between two players, *Editor* and *Author*.
- They both see the same grammar, $\mathcal{G}$ and invalid string $\underset{\sim}{\sigma} \notin \mathcal{L}(\mathcal{G})$.
- Author moves by selecting a single valid string, $\sigma \in \mathcal{L}(\mathcal{G})$.
- Editor moves continuously, sampling a list of $\tilde{\boldsymbol{\sigma}} \in \mathcal{P}(\mathcal{L}(\mathcal{G}))$.
- As soon as Author repairs $\underset{\sim}{\sigma}$, the turn immediately ends.
- Neither player sees the other's move(s) before making their own.
- If Editor anticipates Author's move, i.e., $\sigma \in \tilde{\boldsymbol{\sigma}}$, they both win.
- If Author surprises Editor with a valid move, i.e., $\sigma \notin \tilde{\boldsymbol{\sigma}}$, Author wins.
- We may consider a refinement where Editor wins in proportion to the time taken to anticipate Author's move.

From Error-Correcting Codes to Correcting Coding Errors

> "*Damn it, if the machine can detect an error, why can't it locate the position of the error and correct it?*'"
>
> Richard Hamming, 1915-1998

Bounded Levenshtein Reachability

Syntax repair can be treated as a language intersection problem between a context-free language (CFL) and a regular language.

Definition (Bounded Levenshtein-CFL reachability)

Given a CFL ℓ and an invalid string $\underset{\sim}{\sigma} : \ell^{\complement}$, the BCFLR problem is to find every valid string reachable within d edits of $\underset{\sim}{\sigma}$, i.e., we seek $L(\underset{\sim}{\sigma}, d) \cap \ell$ where $L(\underset{\sim}{\sigma}, d) \coloneqq \{\sigma \mid \Delta(\underset{\sim}{\sigma}, \sigma) \leq d\}$ and Δ is the Levenshtein metric.

To solve this problem, we will first pose a simpler problem that only considers localized edits, then turn our attention back to BCFLR.

Definition (Porous completion)

Let $\underline{\Sigma} \coloneqq \Sigma \cup \{\_\}$, where $\_$ denotes a hole. We denote $\sqsubseteq: \Sigma^n \times \underline{\Sigma}^n$ as the relation $\{\langle \sigma', \sigma \rangle \mid \sigma_i \in \Sigma \implies \sigma'_i = \sigma_i\}$ and the set of all inhabitants $\{\sigma' \mid \sigma' \sqsubseteq \sigma\}$ as $\mathrm{H}(\sigma)$. Given a *porous string*, $\sigma : \underline{\Sigma}^*$ we seek all syntactically admissible inhabitants, i.e., $A(\sigma) \coloneqq \mathrm{H}(\sigma) \cap \ell$.

Ranked repair under realtime constraints

$A(\sigma)$ is often a very large-cardinality set, so we want a procedure which prioritizes likely repairs first, without exhaustive enumeration. Specifically,

Definition (Ranked repair)

Given a finite language $\ell^{\cap} = L(\underset{\sim}{\sigma}, d) \cap \ell$ and a probabilistic language model $P_\theta : \Sigma^* \rightarrow [0, 1] \subset \mathbb{R}$, the ranked repair problem is to find the top-k repairs by likelihood under the language model. That is,

$$R(\ell^{\cap}, P_\theta) \coloneqq_{\{\boldsymbol{\sigma} \mid \boldsymbol{\sigma} \subseteq \ell^{\cap}, |\boldsymbol{\sigma}| \leq k\}} \sum_{\sigma \in \boldsymbol{\sigma}} \prod_{i=1}^{|\sigma|} P_\theta(\sigma_i \mid \sigma_{1\ldots i})^{\frac{1}{|\sigma|}} \tag{1}$$

We want a procedure $\hat{R}$, minimizing $\mathbb{E}_{G,\sigma}\left[D_{\mathrm{KL}}(\hat{R} \parallel R)\right]$ and total latency.

An Simple Reachability Proof

Lemma

For any nonempty language $\ell : \mathcal{L}(\mathcal{G})$ *and invalid string* $\underset{\sim}{\sigma} : \Sigma^n$*, there exists an* $(\tilde{\sigma}, m)$ *such that* $\tilde{\sigma} \in \ell \cap \Sigma^m$ *and* $0 < \Delta(\underset{\sim}{\sigma}, \ell) \leq \max(m, n) < \infty$*, where* Δ *denotes the Levenshtein edit distance.*

Proof.

Since ℓ is nonempty, it must have at least one inhabitant $\sigma \in \ell$. Let $\tilde{\sigma}$ be the smallest such member. Since $\tilde{\sigma}$ is a valid sentence in ℓ, by definition it must be that $|\tilde{\sigma}| < \infty$. Let $m := |\tilde{\sigma}|$. Since we know $\underset{\sim}{\sigma} \notin \ell$, it follows that $0 < \Delta(\underset{\sim}{\sigma}, \ell)$. Let us consider two cases, either $\tilde{\sigma} = \varepsilon$, or $0 < |\tilde{\sigma}|$:

- If $\tilde{\sigma} = \varepsilon$, then $\Delta(\underset{\sim}{\sigma}, \tilde{\sigma}) = n$ by full erasure of $\underset{\sim}{\sigma}$, or
- If $0 < m$, then $\Delta(\underset{\sim}{\sigma}, \tilde{\sigma}) \leq \max(m, n)$ by overwriting.

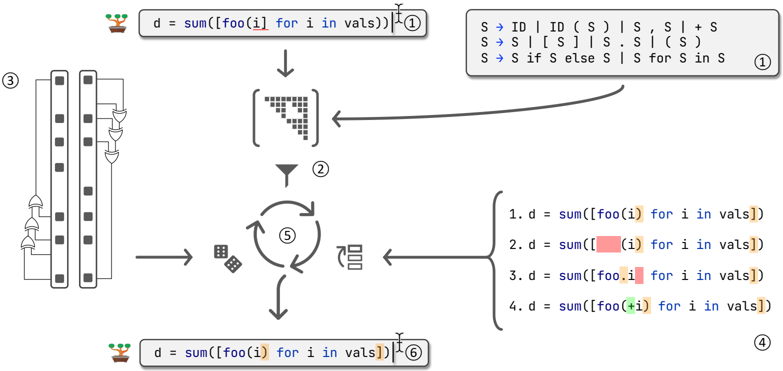
In either case, it follows $\Delta(\underset{\sim}{\sigma}, \ell) \leq \max(m, n)$ and ℓ is always reachable via a finite nonempty set of Levenshtein edits, i.e., $0 < \Delta(\underset{\sim}{\sigma}, \ell) < \infty$. □

From CFL Reachability to Real World Program Repair

To fix real code, we needed to overcome a variety of interesting challenges:

- **Syntax mismatch**: The syntax of real-world programming languages does not exactly correspond to the theory of formal languages.
- **Source code $\approx$ PL**: Most of the time, source code in the wild is incomplete or only loosely approximates a programming language.
- **Responsiveness**: The usefulness of synthetic repairs is inversely proportional to the amount of time required to generate them.
- **Edit generation**: How do we generate edits that are (1) syntactically admissible (2) statistically plausible and (3) semantically meaningful?
- **Evaluation**: Big code and version control is too coarse-grained, contains irrelevant edits, not representative of small errors/fixes.

High-level architecture overview



Our framework consists of three components, (2) a solver, (4) ranker and (3) sampler. Given an invalid string and grammar (1), we first compile them into a multilinear system of equations which can be solved directly, yielding a set of repairs that are ranked using a suitable scoring function (4). Optionally, we may introduce stochastic edits to the string using the Levenshtein ball sampler (3) and extract the solutions incrementally (5).

Uniform sampling benchmark on natural syntax errors

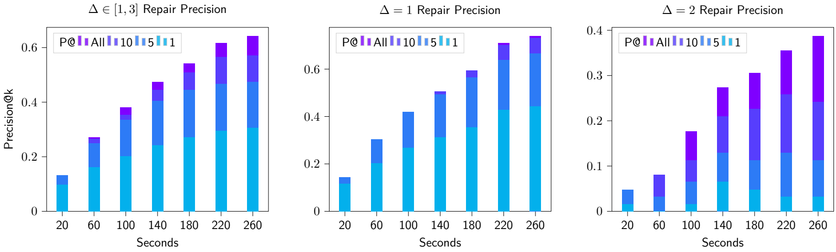


Figure: Repairs discovered before the latency cutoff are reranked based on their tokenwise perplexity and compared for an exact lexical match with the human repair at or below rank k. We note that the uniform sampling procedure is not intended to be used in practice, but provides a baseline for the empirical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

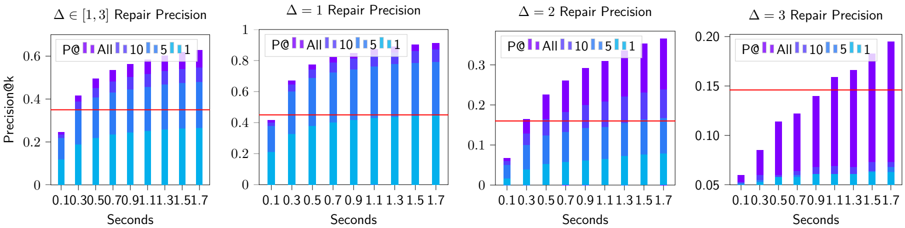


Figure: Adaptive sampling repairs. The red line indicates Seq2Parse precision@1 on the same dataset. Since it only supports generating one repair, we do not report precision@k or the intermediate latency cutoffs.

Precision on Error Correction in Synthetic Language

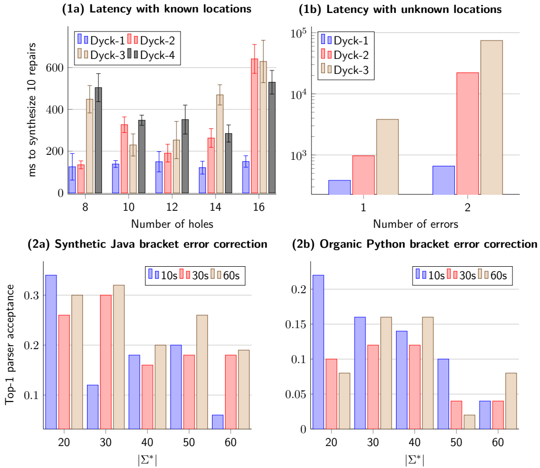


Figure: Benchmarking bracket correction latency and accuracy across two bracketing langauges, one generated from Dyck-n, and the second uses an abstracted source code snippet with imbalanced parentheses.

Uniform sampling benchmark

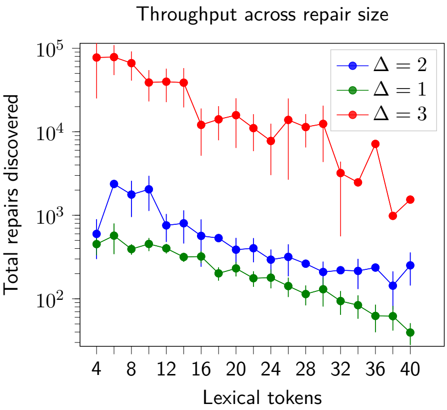
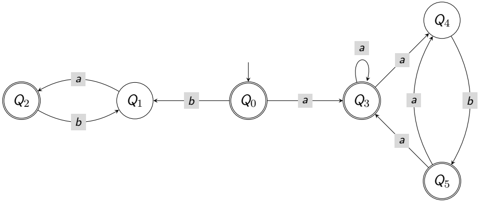


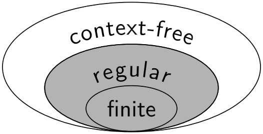
Figure: We evaluate throughput by sampling edits across invalid strings $|\underset{\sim}{\sigma}| \leq 40$ from the StackOverflow dataset of varying length, and measure the total number of syntactically valid edits discovered, as a function of string length and language edit distance $\Delta \in [1,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

Background: Regular grammars

A regular grammar (RG) is a quadruple $\mathcal{G} = \langle V, \Sigma, P, S \rangle$ where V are nonterminals, Σ are terminals, $P : V \times (V \cup \Sigma)^{\leq 2}$ are the productions, and $S \in V$ is the start symbol, i.e., all productions are of the form $A \rightarrow a$, $A \rightarrow aB$ (right-regular), or $A \rightarrow Ba$ (left-regular). E.g., the following RG and NFA correspond to the language defined by the *regex* `(a(ab)*)*(ba)*`:

$$
\begin{aligned}
S &\rightarrow Q_0 \mid Q_2 \mid Q_3 \mid Q_5 \\
Q_0 &\rightarrow \varepsilon \\
Q_1 &\rightarrow Q_0 b \mid Q_2 b \\
Q_2 &\rightarrow Q_1 a \\
Q_3 &\rightarrow Q_0 a \mid Q_3 a \mid Q_5 a \\
Q_4 &\rightarrow Q_3 a \mid Q_5 a \\
Q_5 &\rightarrow Q_4 b
\end{aligned}
$$



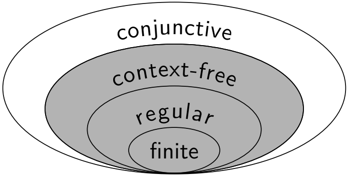


Background: Context-free grammars

In a context-free grammar $\mathcal{G} = \langle V, \Sigma, P, S \rangle$ all productions are of the form $P : V \times (V \cup \Sigma)^+$, i.e., RHS may contain any number of nonterminals, V. Recognition decidable in n^ω, n.b. CFLs are **not** closed under intersection!

For example, consider the grammar $\underline{S \rightarrow SS \mid (S) \mid ()}$. This represents the language of balanced parentheses, e.g. $(), ()(), (()), ()(()), (()()), (())() \ldots$

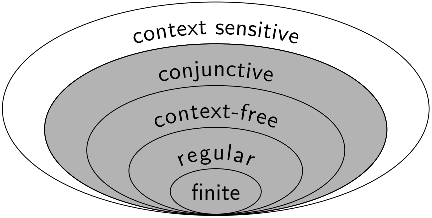
Every CFG has a normal form $P^* : V \times (V^2 \mid \Sigma)$, i.e., every production can be refactored into either $v_0 \rightarrow v_1 v_2$ or $v_0 \rightarrow \sigma$, where $v_{0\ldots2} : V$ and $\sigma : \Sigma$, e.g., $\{S \rightarrow SS \mid (S) \mid ()\} \Leftrightarrow^* \{S \rightarrow XR \mid SS \mid LR, L \rightarrow (, R \rightarrow), X \rightarrow LS\}$



Background: Conjunctive grammars

Conjunctive grammars naturally extend CFGs with CFL union and intersection, respecting closure under those operations. Equivalent to trellis automata, which are like contractive elementary cellular automata. Language inclusion is decidable in P.

$$\frac{\Gamma \vdash \mathcal{G}_1, \mathcal{G}_2 : \texttt{CG}}{\Gamma \vdash \exists\, \mathcal{G}_3 : \texttt{CG} \,.\, \mathcal{L}_{\mathcal{G}_1} \cap \mathcal{L}_{\mathcal{G}_1} \leftrightarrow \mathcal{G}_3} \cap$$



Background: Closure properties of formal languages

Formal languages are not always closed under set-theoretic operations, e.g., CFL $\cap$ CFL is not CFL in general. Let $\cdot$ denote concatenation, $\star$ be Kleene star, and $\complement$ be complementation:

	$\cup$	$\cap$	$\cdot$	$\star$	$\complement$
Finite[1]	✓	✓	✓	✓	✓
Regular[1]	✓	✓	✓	✓	✓
Context-free[1]	✓	✗	✓	✓	✗
Conjunctive[1,2]	✓	✓	✓	✓	?
Context-sensitive[2]	✓	✓	✓	+	✓
Recursively Enumerable[2]	✓	✓	✓	✓	✗

We would like a language family that is (1) tractable, i.e., has polynomial recognition and search complexity and (2) reasonably expressive, i.e., can represent syntactic properties of real-world programming languages.

Context-free parsing, distilled

Given a CFG $\mathcal{G} := \langle V, \Sigma, P, S \rangle$ 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:

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

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

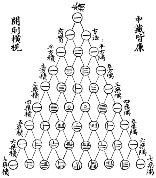
If we define $\sigma_r^{\nrightarrow} := \{w \mid (w \to \sigma_r) \in P\}$, then initialize $M^0_{r+1=c}(\mathcal{G}', e) := \sigma_r^{\nrightarrow}$ and solve for the fixpoint $M^* = M + M^2$,

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

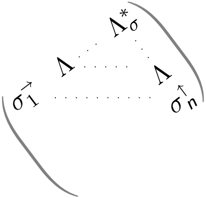
Valiant (1975) shows that $\sigma \in \mathcal{L}_{\mathcal{G}}$ iff $S \in \mathcal{T}$, i.e., $\mathbb{1}_{\mathcal{T}}(S) \iff \mathbb{1}_{\mathcal{L}_{\mathcal{G}}}(\sigma)$.

Lattices, Matrices and Trellises

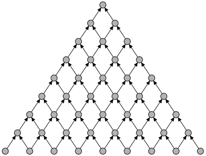
The art of treillage has been practiced from ancient through modern times.



Jia Xian Triangle
Jia, ∼1030 A.D.



CYK Parsing
Sakai, 1961 A.D.

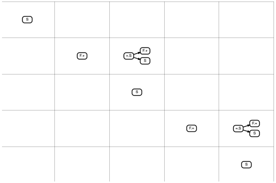


Trellis Automaton
Dyer, 1980 A.D.

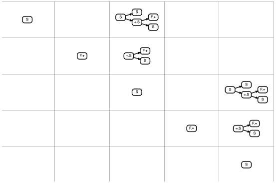
A few observations on algebraic parsing

- The matrix $\mathbf{M}^*$ is strictly upper triangular, i.e., nilpotent of degree n
- Recognizer can be translated into a parser by storing backpointers

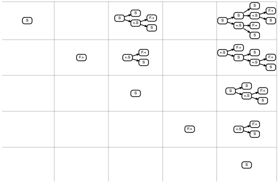
$\mathbf{M}_1 = \mathbf{M}_0 + \mathbf{M}_0^2$



$\mathbf{M}_2 = \mathbf{M}_1 + \mathbf{M}_1^2$



$\mathbf{M}_3 = \mathbf{M}_2 + \mathbf{M}_2^2 = \mathbf{M}_4$



- The $\otimes$ operator is *not* associative: $S \otimes (S \otimes S) \neq (S \otimes S) \otimes S$
- Built-in error recovery: nonempty submatrices = parsable fragments
- `seekFixpoint { it + it * it }` is sufficient but unnecessary
- If we had a way to solve for $\mathbf{M} = \mathbf{M} + \mathbf{M}^2$ directly, power iteration would be unnecessary, could solve for $\mathbf{M} = \mathbf{M}^2$ above superdiagonal

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. $\mathbb{1}_{\subseteq V}: V \to \mathbb{Z}_2^{|V|}$
- $\oplus, \otimes$ are defined as $\boxplus, \boxtimes$, so that the following diagram commutes:

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

- 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
- More sophisticated representations are known for $\binom{n}{0\leq k}$ subsets
- If density is desired, possible to use the Maculay representation

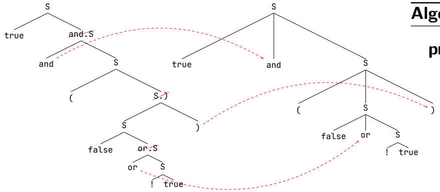
Satisfiability + holes (our contribution)

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_∞	$\begin{pmatrix} \{N\} & \varnothing & \{S\} \\ & \{N,O\} & \{L\} \\ & & \{N,O\} \end{pmatrix}$	$\begin{pmatrix} \square\blacksquare\square\square & \square\square\square\square & \square\square\square\blacksquare \\ & \square\blacksquare\blacksquare\square & \blacksquare\square\square\square \\ & & \square\blacksquare\blacksquare\square \end{pmatrix}$	$\begin{pmatrix} V_{0,1} & V_{0,2} & V_{0,3} \\ & V_{1,2} & V_{1,3} \\ & & V_{2,3} \end{pmatrix}$

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

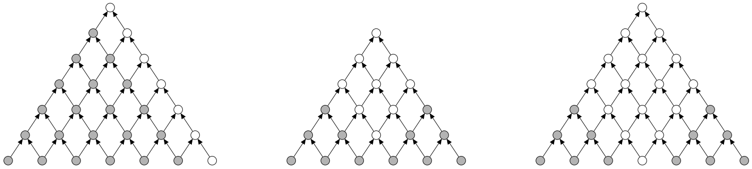
Algorithm Rewrite procedure for tree denormalization

procedure $\textsc{Denormalize}(\texttt{t: Tree})$
  $\texttt{stems} \leftarrow \{\, \textsc{Denormalize}(\texttt{c}) \mid \texttt{c} \in \texttt{t.children} \,\}$
  if $\texttt{t.root} \in V_{\mathcal{G}'} \setminus V_{\mathcal{G}}$ **then**
    return $\texttt{stems}$ ▷ Drop synthetic nonterminals.
  else ▷ Graft the denormalized children on root.
    return $\{\, \texttt{Tree(root, stems)} \,\}$
  end if
end procedure

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

Incremental parsing

Should only need to recompute submatrices affected by individual edits. In the worst case, each edit requires quadratic complexity in terms of $|\Sigma^*|$, assuming $\mathcal{O}(1)$ cost for each CNF-nonterminal subset join, $\mathbf{V}'_1 \otimes \mathbf{V}'_2$.



Append $\mathcal{O}(n+1)$

Delete $\mathcal{O}\left(\frac{1}{4}(n-1)^2\right)$

Insert $\mathcal{O}\left(\frac{1}{4}(n+1)^2\right)$

Related to **dynamic matrix inverse** and **incremental transitive closure** with vertex updates. With a careful encoding, we can incrementally update SAT constraints as new keystrokes are received to eliminate redundancy.

Conjunctive parsing

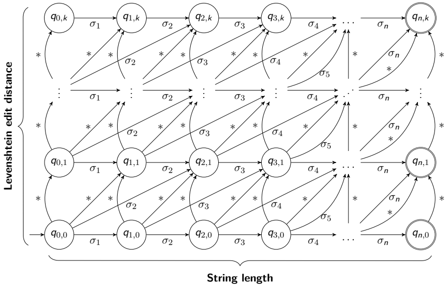
It is well-known that the family of CFLs is not closed under intersection. For example, consider $\mathcal{L}_\cap := \mathcal{L}_{\mathcal{G}_1} \cap \mathcal{L}_{\mathcal{G}_2}$:

$$P_1 := \{\ S \to LR, \quad L \to ab \mid aLb, \quad R \to c \mid cR\ \}$$
$$P_2 := \{\ S \to LR, \quad R \to bc \mid bRc, \quad L \to a \mid aL\ \}$$

Note that $\mathcal{L}_\cap$ generates the language $\{\ a^d b^d c^d \mid d > 0\ \}$, which according to the pumping lemma is not context-free. To encode $\mathcal{L}_\cap$, we intersect all terminals $\Sigma_\cap := \bigcap_{i=1}^{c} \Sigma_i$, then for each $t_\cap \in \Sigma_\cap$ and CFG, construct an equivalence class $E(t_\cap, \mathcal{G}_i) = \{w_i \mid (w_i \to t_\cap) \in P_i\}$ as follows:

$$\bigwedge_{t \in \Sigma_\cap} \bigwedge_{j=1}^{c-1} \bigwedge_{i=1}^{|\sigma|} E(t_\cap, \mathcal{G}_j) \equiv_{\sigma_i} E(t_\cap, \mathcal{G}_{j+1}) \tag{2}$$

MAIA



CFG

$$
\begin{aligned}
S &\Rightarrow \{\cdot \in Q \mid \delta(\cdot, q_{n,0}) \leq k\}\\
* &\Rightarrow \{\cdot \in \Sigma\}
\end{aligned}
$$

$$\big\{q_{i,j} \Rightarrow \{q_{i,j-1}*\} \mid i,j \in [1,n] \times [1,k]\big\}$$

$$\big\{q_{i,j} \Rightarrow \{q_{i-1,j-1}*\} \mid i,j \in [1,n] \times [1,k]\big\}$$

$$\big\{q_{i,j} \Rightarrow \{q_{i-1,j}\sigma_i\} \mid i,j \in [1,n] \times [0,k]\big\}$$

$$\big\{q_{i,j} \Rightarrow \{q_{i-2,j-1}\sigma_i\} \mid i,j \in [2,n] \times [1,k]\big\}$$

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.

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

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

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

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

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

Semiring algebras: Part I

There are a number of alternate semirings which can be used to solve for $A(\sigma)$. A naïve approach accumulates a mapping of nonterminals to sets of strings. Letting $D = V \rightarrow \mathcal{P}(\Sigma^*)$, we define $\oplus, \otimes : D \times D \rightarrow D$. Initially, we construct $M_0[r+1=c] = p(\sigma_r)$ using:

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

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

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

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

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:

$$X \oplus Z \mapsto \bigcup_{k \in \pi_1(X \cup Z)} \big\{k \Rightarrow \mathbb{T}_2(k, Q_x \cup Q_z) \mid Q_x \in \pi_2(X \circ k), Q_z \in \pi_2(Z \circ k)\big\}$$

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

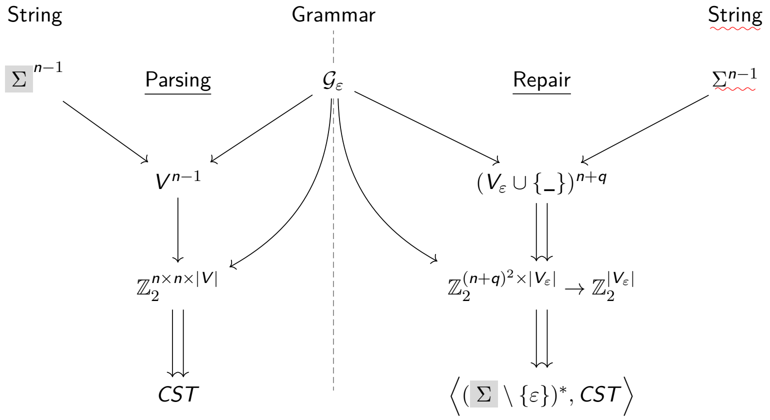
A birds eye view of the algorithm

We can lower Valiant's algorithm onto a polynomial system of equations over finite fields, allowing us to solve for holes and parse trees.

$$
\begin{tikzcd}
\langle \mathcal{G}', \boxed{\Sigma}^{n-1} \rangle \arrow[dr, leftrightarrow] & \text{Set} & \text{Bit} & \langle \mathcal{G}', \Sigma^{n-1} \rangle \arrow[dr] & \text{SAT} \\
\text{Rubix} & M \times M \arrow[r, leftrightarrow, "\mathbb{1}^{2^{n\times n}}", "\mathbb{1}^{-2^{n\times n}}"'] \arrow[d, "\oplus"', "\otimes"] & \mathbb{Z}_2^{|V|^{n\times n}} \times \mathbb{Z}_2^{|V|^{n\times n}} \arrow[rr, tail, "\varphi^{2^{n\times n}}", "\varphi^{-2^{n\times n}}"'] \arrow[d, "\boxed{+}"', "\boxed{*}"] & & \mathcal{M} \times \mathcal{M} \arrow[d, leftrightarrow, "+"', "*"] \\
\text{Matrix} & 2^V \times 2^V \arrow[r, leftrightarrow, "\mathbb{1}^{2}", "\mathbb{1}^{-2}"'] \arrow[d, "\oplus"', "\otimes"] & \mathbb{Z}_2^{|V|} \times \mathbb{Z}_2^{|V|} \arrow[rr, tail, "\varphi^{2}", "\varphi^{-2}"'] \arrow[d, "\boxed{\boxplus}"', "\boxed{\boxtimes}"] & & \mathcal{V} \times \mathcal{V} \arrow[d, leftrightarrow, "\boxplus"', "\boxtimes"] \\
\text{Vector} & 2^V \arrow[r, leftrightarrow, "\mathbb{1}", "\mathbb{1}^{-1}"'] & \mathbb{Z}_2^{|V|} \arrow[rr, tail, "\varphi", "\varphi^{-1}"'] & & \mathcal{V}
\end{tikzcd}
$$

So far, we only consider Cartesian closed categories, however, we can also consider other categories, such as the category of CFLs under conjunction, which allows us to encode the intersection of two CFGs.

A birds eye view of the algorithm



Our algorithm produces set of concrete syntax trees (CSTs) for a given valid string. Otherwise, if the string contains an error, the algorithm generates a set of admissible corrections, alongside their CSTs.

Error Correction: Levenshtein q-Balls

Now that we have a reliable method to fix *localized* errors, $S : \mathcal{G} \times (\Sigma \cup \{\varepsilon, \_\})^n \rightarrow \{\Sigma^n\} \subseteq \mathcal{L}_\mathcal{G}$, given some unparseable string, i.e., $\sigma_1 \ldots \sigma_n : \boxed{\Sigma}^n \cap \mathcal{L}_\mathcal{G}^\complement$, where should we put holes to obtain a parseable $\sigma' \in \mathcal{L}_\mathcal{G}$? One way to do so is by sampling repairs, $\boldsymbol{\sigma} \sim \Sigma^{n \pm q} \cap \Delta_q(\sigma)$ from the Levenshtein q-ball centered on σ, i.e., the space of all admissible edits with Levenshtein distance $\leq q$ (this is loosely analogous to a finite difference approximation). To admit variable-length edits, we first add an ε^+-production to each unit production:

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

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

Error Correction: d-Subset Sampling

Next, suppose $U : \mathbb{Z}_2^{m \times m}$ is a matrix whose structure is shown in Eq. 3, wherein C is a primitive polynomial over $\mathbb{Z}_2^m$ with coefficients $C_{1\ldots m}$ and semiring operators $\oplus := \veebar, \otimes := \land$:

$$U^t V = \begin{pmatrix} C_1 & \cdots & \cdots & \cdots & C_m \\ \top & \circ & \cdots & \cdots & \circ \\ \circ & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \circ & \cdots & \circ & \top & \circ \end{pmatrix}^t \begin{pmatrix} V_1 \\ \vdots \\ \vdots \\ \vdots \\ V_m \end{pmatrix} \tag{3}$$

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 $\left\{{n \atop d}\right\} \times \Sigma_\varepsilon^{2d}$.

Error Correction: Sketch Templates

Finally, to sample $\boldsymbol{\sigma} \sim \Delta_q(\underset{\sim}{\sigma})$, we enumerate a series of sketch templates $H(\sigma, i) = \sigma_{1\ldots i-1}\ \_\ \_\ \sigma_{i+1\ldots n}$ 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:

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

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

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

A pairing function for sampling bounded binary trees

The type $\mathbb{T}_2$ of all trees that can be generated by a CNF CFG is a nested datatype: $P(a) = 1 + aL(P(a)^2), L(p) = 1 + pL(p)$. We can count and sample the total number of trees induced by a given sketch template as:

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

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

Probabilistic repair generation

Algorithm Probabilistic reachability

Require: $\mathcal{G}$ grammar, $\underset{\sim}{\sigma}$ broken string, p process ID, c total CPU cores, t_{total} timeout.

```
Q ← ∅, R ← ∅, ε ← 1, i ← 0, Y ~ Z_2^m, t_0 ← t_now        ▷ Initialize replay buffer Q and reservoir R.
repeat
    if Q = ∅ or Rand(0, 1) < ε then
        σ̂ ← φ^{-1}(⟨κ, ρ⟩^{-1}(U^{ci+p} Y), σ), i ← i + 1    ▷ Sample WoR using the leapfrog method.
    else
        σ̂ ~ Q + Noise(Q)                                      ▷ Sample replay buffer with additive noise.
    end if
    R ← R ∪ {σ̂}                                               ▷ Insert repair candidate σ̂ into reservoir R.
    if R is full then
        σ̂ ←_{σ̂∈R} PP(σ̂)                                      ▷ Select lowest perplexity repair candidate.
        if σ̂ ∈ L(G) then
            Q ← Q ∪ {σ̂}                                       ▷ Insert successful repair into replay buffer.
        end if
        R ← R \ {σ̂}                                           ▷ Remove checked sample from the reservoir.
    end if
    ε ← Schedule((t_now − t_0)/t_total)                       ▷ Update exploration/exploitation rate.
until t_total elapses.
return σ̃ ∈ Q ranked by PP(σ̃).
```

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**

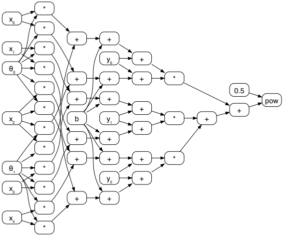
Classical programs are graphs

Programs can be compiled into DFGs and represented using a big matrix.

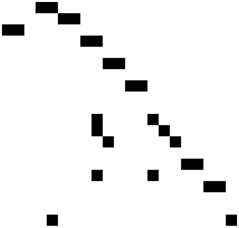
Program

```
sum = 0
l = [0, 0, 0, 0]
for i in range(0, 4):
 l[i] += θ[i] * x[i]
for i in range(0, 4):
 l[i] -= y[i] - b
for i in range(0, 4):
 l[i] *= l[i]
for i in range(0, 4):
 sum += l[i]
l = sqrt(sum)
```

Dataflow Graph



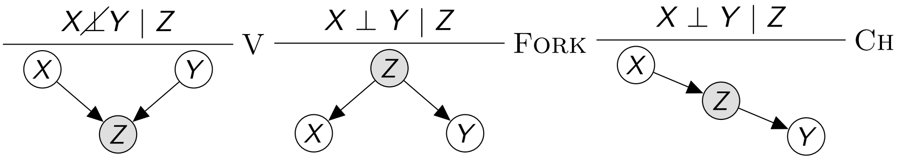
Matrix



This representation allows us to solve for their fixedpoints as eigenvectors.

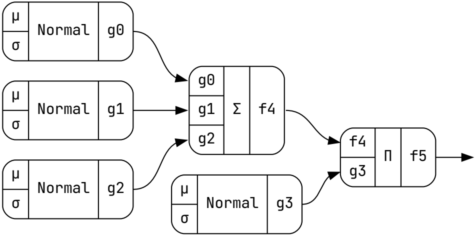
Probabilistic programs are also graphs

A Bayesian Belief Network (BN) is an acyclic DGM of the following form:



Translatable to a probabilistic circuit a.k.a. Sum Product Network (SPN):

$$
\begin{aligned}
PC &\to v \sim \mathcal{D} \\
PC &\to PC \oplus PC \\
PC &\to PC \otimes PC
\end{aligned}
$$



Message passing & path algebras

A semiring algebra, denoted $(S, \oplus, \otimes, ⓪, ①)$, is a set together with two binary operators $\oplus, \otimes : S \times S \to S$ such that $(S, \oplus, ⓪)$ is a commutative monoid and $(S, \otimes, ①)$ is a monoid. Furthermore, we have distributivity:

$$\frac{a \cdot (b \cdot c)}{(a \cdot b) \cdot c}\ \textsc{Assoc} \qquad \frac{a \cdot ①}{a}\ \textsc{Neutral} \qquad \frac{a \cdot b}{b \cdot a}\ \textsc{Comm}$$

$$\frac{(a \oplus b) \otimes c}{(a \otimes c) \oplus (b \otimes c)}\ \textsc{Dist} \qquad \frac{a \otimes ⓪}{⓪}\ \textsc{Annhil}$$

These operators can be lifted to matrices to form *path algebras*:

$$\delta_{st} = \overbrace{\bigoplus_{P \in P^*_{st}} \underbrace{\bigotimes_{e \in P}}_{\text{Aggregate}} W_e}^{\text{Update}}$$

$\oplus$	$\otimes$	⓪	①	Path
min	$+$	∞	0	Shortest
max	$+$	$-\infty$	0	Longest
max	min	0	∞	Widest
$\veebar$	$\wedge$	$\circ$	$\top$	Random

Recap: Classical logic in a nutshell

$$\frac{a \veebar b}{(p \vee q) \wedge \neg(p \wedge q)}\text{XOR} \qquad \frac{a \to b}{\neg a \vee b}\text{Impl}$$

$$\frac{a \leftrightarrow b}{(\neg a \vee b) \wedge (\neg b \vee a)}\text{Iff}$$

$$\frac{\neg\neg a}{a}\text{2Neg} \qquad \frac{a \cdot (b \cdot c)}{(a \cdot b) \cdot c}\text{Assoc}_{\wedge\vee} \qquad \frac{a \cdot b}{b \cdot a}\text{Comm}_{\wedge\vee}$$

$$\frac{a \wedge (b \vee c)}{(a \wedge b) \vee (a \wedge c)}\text{Dist}_{\wedge} \qquad \frac{a \vee (b \wedge c)}{(a \vee b) \wedge (a \vee c)}\text{Dist}_{\vee}$$

$$\frac{\neg(a \vee b)}{\neg a \wedge \neg b}\text{DeMorgan}_{\vee} \qquad \frac{\neg(a \wedge b)}{\neg a \vee \neg b}\text{DeMorgan}_{\wedge}$$

Normalization in classical logic

Conjunctive Normal Form

$$\textsc{Conj} \rightarrow (\textsc{Disj}) \mid \textsc{Conj} \wedge (\textsc{Disj})$$
$$\textsc{Unit} \rightarrow \textsc{Var} \mid \neg\textsc{Var} \mid \bot \mid \top$$
$$\textsc{Disj} \rightarrow \textsc{Unit} \mid \textsc{Disj} \vee \textsc{Disj}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(x \vee \neg y) \vee \neg\neg z}{\neg(x \vee \neg y) \vee z}\text{ 2Neg}}{(\neg x \wedge \neg\neg y) \vee z}\text{ DeMorgan}}{(\neg x \wedge y) \vee z}\text{ 2Neg}}{(\neg x \vee z) \wedge (y \vee z)}\text{ Dist}$$

Zhegalkin Normal Form

$$f(x_1, \ldots x_n) = \bigoplus_{i \subseteq \{1,\ldots,n\}} a_i x^i$$

i.e., a_i's filter the powerset.

$$\dfrac{\dfrac{\dfrac{\dfrac{x + (y \wedge \neg z)}{x + y(1 \oplus z)}}{x + (y \oplus yz)}}{x \oplus (y \oplus yz) \oplus x(y \oplus yz)}}{x \oplus y \oplus xy \oplus yz \oplus xyz}$$

Some common algebraic and logical forms

a_1	a_2	a_3	a_4	ZNF	Logical	CNF
0	0	0	0	0	$\bot$	$x \wedge \neg x$
1	0	0	0	1	$\top$	$x \vee \neg x$
0	1	0	0	x	x	x
1	1	0	0	1 + x	$\neg x$	$\neg x$
0	0	1	0	y	y	y
1	0	1	0	1 + y	$\neg y$	$\neg y$
0	1	1	0	x + y	$x \oplus y$	$(x \vee y) \wedge (\neg x \vee \neg y)$
1	1	1	0	1 + x + y	$x \iff y$	$(x \vee \neg y) \wedge (\neg x \vee y)$
0	0	0	1	xy	$x \wedge y$	$x \wedge y$
1	0	0	1	1 + xy	$\neg(x \wedge y)$	$(\neg x) \vee (\neg y)$
0	1	0	1	x + xy	$x \wedge (\neg y)$	$x \wedge (\neg y)$
1	1	0	1	1 + x + xy	$x \implies y$	$(\neg x) \vee y$
0	0	1	1	y + xy	$(\neg x) \wedge y$	$(\neg x) \wedge y$
1	0	1	1	1 + y + xy	$x \impliedby y$	$x \vee (\neg y)$
0	1	1	1	x + y + xy	$x \vee y$	$x \vee y$
1	1	1	1	1 + x + y + xy	$\neg(x \vee y)$	$(\neg x) \wedge (\neg y)$

Facts about finite fields

- For every prime number p and positive integer n, there exists a finite field with p^n elements, denoted $GF(p^n)$, $\mathbb{Z}/p^n$ or $\mathbb{F}_p^n$.
- The following instruction sets have identical expressivity:
  - Pairs: $\{\vee, \neg\}, \{\wedge, \neg\}, \{\rightarrow, \neg\}, \{\rightarrow, \bot\}, \{\rightarrow, \veebar\}, \{\wedge, \veebar\}, \ldots$
  - Triples: $\{\vee, =, \veebar\}, \{\vee, \veebar, \top\}, \{\wedge, =, \bot\}, \{\wedge, =, \veebar\}, \{\wedge, \veebar, \top\}, \ldots$
- In other words, we can compute any Boolean function $\mathbb{B}^n \rightarrow \mathbb{B}$ by composing any one of the above operator sets in an orderly fashion.
- $\mathbb{F}_2$ corresponds to arithmetic modulo 2, i.e., $\oplus := \veebar, \otimes := \wedge$.
- There are (at least) two schools of thought about Boolean circuits:
  - Logical: Conjunctive Normal Form (CNF). May not be unique.
  - Algebra: Zhegalkin Normal Form (ZNF). Always unique.
- The type $\mathbb{F}_2^n \rightarrow \mathbb{F}_2$ possesses 2^{2^n} inhabitants.

Preface to “Two Memoirs on Pure Analysis”

“*Long algebraic calculations were at first hardly necessary for mathematical progress... It was only since Euler that concision has become indispensable to continuing the work this great geometer has given to science. Since Euler, calculation has become more and more necessary and... the algorithms so complicated that progress would be nearly impossible without the elegance that modern geometers have brought to bear on their research, and by which means the mind can promptly and with a glance grasp a large number of operations.*

...

It is clear that elegance, so admirably and aptly named, has no other purpose.

...

Jump headlong into the calculations! Group the operations, classify them by their difficulties and not their appearances. This, I believe, is the mission of future geometers. This is the road on which I am embarking in this work.”

Évariste Galois, 1811-1832

What's the point?

- Algebraists have developed a powerful language for rootfinding
- Tradition handed down from Fermat, Euler, Galois, Kleene, Chomsky
- We have closed forms for exponentials of structured matrices
- Solving these forms can be much faster than power iteration
- Unifies many problems in PL, probability and graph theory
- Context-free parsing is just rootfinding on a semiring algebra
- Unification/simplification a form of lazy hypergraph search
- Bounded program synthesis is matrix factorization/completion
- By doing so, we can leverage well-known algebraic techniques

Future work

Parsing

- Error propagation in discrete dynamical systems and TRS
- Dynamic matrix inverse and incremental transitive closure
- Language Edit Distance with metrizable Boolean semirings
- Unify parser-lexer for scannerless ECP on a real language
- Investigate the feasibility of grammar induction and repair
- Strengthen the connection to Leibnizian differentiability

Probability

- Look into Markov chains (detailed balance, stationarity, reversibility)
- Fuse Valiant parser and probabilistic context-free grammar
- Contextualize belief propagation and graph diffusion processes
- Look into constrained optimization (e.g., L/QP) to rank feasible set

Special thanks

Jin Guo, Xujie Si
Brigitte Pientka, David Yu-Tung Hui,
Ori Roth, Younesse Kaddar, Michael Schröder
Torsten Scholak, Matthew Sotoudeh, Paul Zhu

Learn more at:

https://tidyparse.github.io