A Word Sampler for Well-Typed Functions

Breandan Mark Considine

January 21, 2026

Formal langauges & type theory

$$\underbrace{\sigma \in \mathcal{L}(G) \Leftrightarrow \exists V.\ V \Rightarrow^*_G \sigma}_{\text{membership / parse tree}} \quad \leftrightsquigarrow \quad \underbrace{\exists \tau.\ (\Gamma \vdash e : \tau)}_{\text{type checking / proof tree}}$$

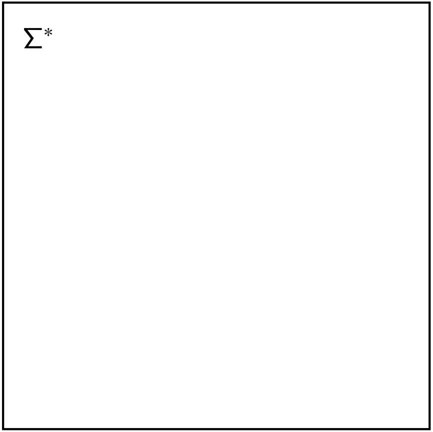
$$\underbrace{(W \to XZ) \in P}_{\text{grammar production}} \quad \leftrightsquigarrow \quad \underbrace{\frac{\Gamma \vdash x : X \qquad \Gamma \vdash z : Z}{\Gamma \vdash xz : W}}_{\text{typing judgment}}$$

$$\underbrace{\mathcal{L}(G) \neq \varnothing \Leftrightarrow \exists \sigma.\ S \Rightarrow^*_G \sigma}_{\text{non-emptiness / generation}} \quad \leftrightsquigarrow \quad \underbrace{\exists e.\ (\Gamma \vdash e : \tau)}_{\text{type inhabitation / synthesis}}$$

Goal: Given a set of typing judgements and a typing context (Γ), design a grammar, G, s.t. $\forall \sigma \in \Sigma^{<n} \exists \tau\ .\ \sigma \in \mathcal{L}(G) \iff \Gamma \vdash \sigma : \tau$.

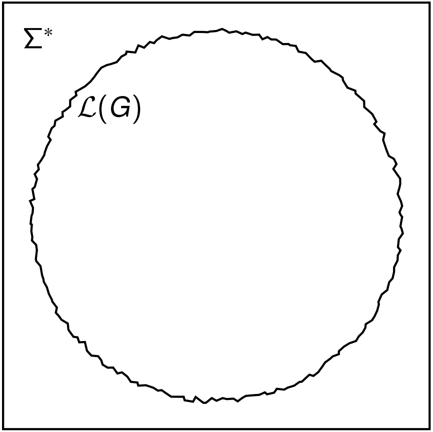
Programming language [in]approximability

- Σ^*: all words over Σ



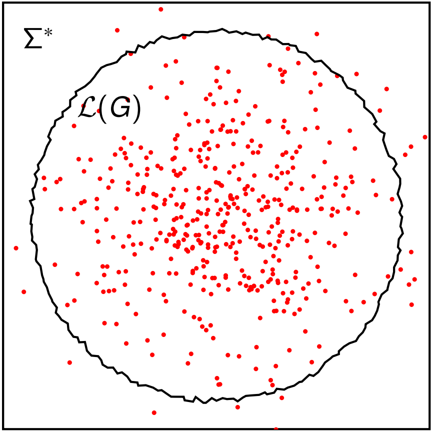
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid



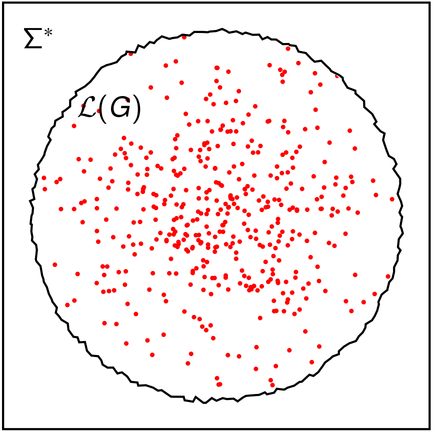
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid
- Most LLMs: $\sigma \leftarrow\!\!\!\rightsquigarrow \Sigma^*$



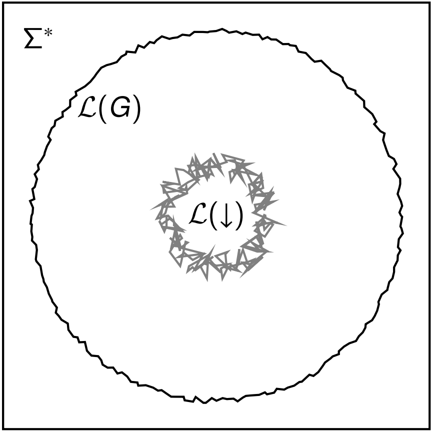
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid
- Most LLMs: $\sigma \leftsquigarrow \Sigma^*$
- Guidance: $\sigma \leftsquigarrow \mathcal{L}(G)$



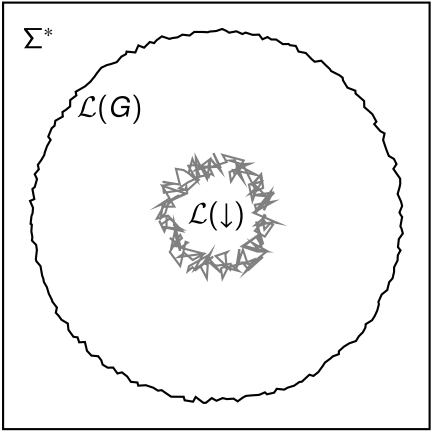
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid
- Most LLMs: $\sigma \leftsquigarrow \Sigma^*$
- Guidance: $\sigma \leftsquigarrow \mathcal{L}(G)$
- $\mathcal{L}(\downarrow)$: halting programs



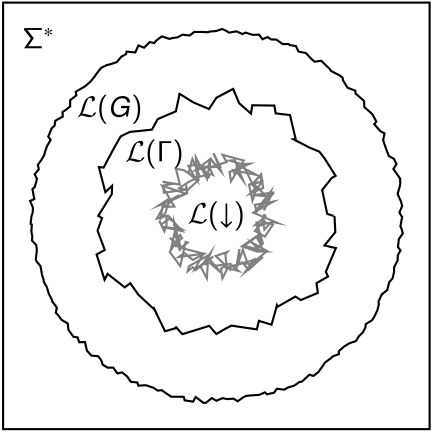
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid
- Most LLMs: $\sigma \leftsquigarrow \Sigma^*$
- Guidance: $\sigma \leftsquigarrow \mathcal{L}(G)$
- $\mathcal{L}(\downarrow)$: halting programs
- Tighter approximations require ever-increasing expressive power



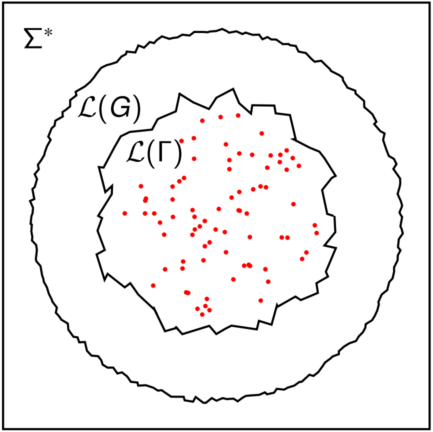
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid
- Most LLMs: $\sigma \leftsquigarrow \Sigma^*$
- Guidance: $\sigma \leftsquigarrow \mathcal{L}(G)$
- $\mathcal{L}(\downarrow)$: halting programs
- Tighter approximations require ever-increasing expressive power
- $\mathcal{L}(\Gamma)$: type-safe programs



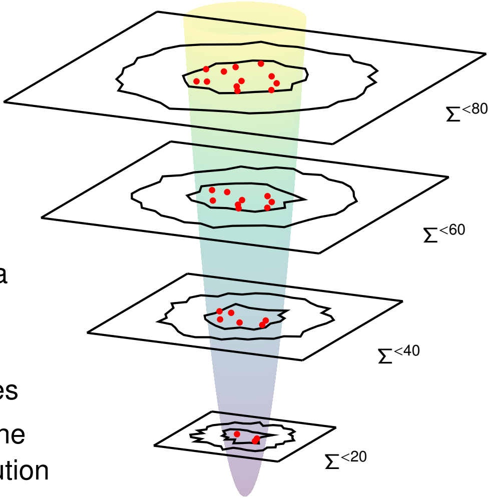
Programming language [in]approximability

- Σ^*: all words over Σ
- $\mathcal{L}(G)$: syntactically valid
- Most LLMs: $\sigma \leftsquigarrow \Sigma^*$
- Guidance: $\sigma \leftsquigarrow \mathcal{L}(G)$
- $\mathcal{L}(\downarrow)$: halting programs
- Tighter approximations require ever-increasing expressive power
- $\mathcal{L}(\Gamma)$: type-safe programs
- Typesafe: $\sigma \leftsquigarrow \mathcal{L}(\Gamma)$



Stratified sampling with finite model theory

- But $\mathcal{L}(\Gamma)$ is infinite
- Consider finite models
- Isolate key complexity parameters of interest
- Embed description into a context-free grammar
- Disintegrate into fixed-parameter tractable slices
- Sample uniformly from the exact conditional distribution



High-level grammar embedding recipe

- Fix a finite type universe $\mathbb{T}$ and an ambient global context Γ
- Decorate vanilla nonterminals with a typing annotation, $\mathtt{E}[\tau]$
- Each typing judgment becomes a schema for constructing a family of synthetic productions, each instantiated with $\tau : \mathbb{T}$

Syntax: $$\frac{\Gamma \vdash e_1 : \tau_1, \cdots, e_m : \tau_m \qquad \Phi(\Sigma, \tau_1, \cdots, \tau_m) : \tau}{\big(\mathtt{E}[\tau] \to \Phi(\Sigma, \tau_1, \cdots, \tau_m)\big) \in P_\Gamma}$$

Names: $$\Gamma \vdash e : \tau \;\Rightarrow\; \big(\mathtt{E}[\tau] \to \mathtt{e}\big) \in P_\Gamma$$

Functions: $$\frac{\Gamma \vdash f : (\tau_1, \cdots, \tau_k) \to \tau}{\big(\mathtt{E}[\tau_1] \to \mathtt{f}\ \mathtt{(}\ \mathtt{E}[\tau_1]\ \mathtt{,}\ \cdots\ \mathtt{,}\ \mathtt{E}[\tau_k]\ \mathtt{)}\big) \in P_\Gamma}$$

Example language: simply typed function syntax

$$
\begin{array}{lcl}
\mathtt{FUN} & ::= & \mathtt{fun}\ \mathtt{f0}\ \mathtt{(}\ \mathtt{PRM}\ \mathtt{)}\ \mathtt{:}\ \mathbb{T}\ \mathtt{=}\ \mathtt{EXP} \\
\mathtt{PRM} & ::= & \mathtt{PID}\ \mathtt{:}\ \mathbb{T} \mid \mathtt{PRM}\ \mathtt{,}\ \mathtt{PID}\ \mathtt{:}\ \mathbb{T} \\
\mathtt{EXP} & ::= & \ulcorner\mathbb{N}\lrcorner \mid \ulcorner\mathbb{B}\lrcorner \mid \mathtt{PID} \mid \mathtt{INV} \mid \mathtt{IFE} \mid \mathtt{OPX} \\
\mathtt{OPX} & ::= & \mathtt{(}\ \mathtt{EXP}\ \mathtt{OPR}\ \mathtt{EXP}\ \mathtt{)} \\
\mathtt{IFE} & ::= & \mathtt{if}\ \mathtt{EXP}\ \mathtt{\{}\ \mathtt{EXP}\ \mathtt{\}}\ \mathtt{else}\ \mathtt{\{}\ \mathtt{EXP}\ \mathtt{\}} \\
\mathtt{INV} & ::= & \mathtt{FID}\ \mathtt{(}\ \mathtt{ARG}\ \mathtt{)} \\
\mathtt{ARG} & ::= & \mathtt{EXP} \mid \mathtt{ARG}\ \mathtt{,}\ \mathtt{EXP} \\
\mathtt{OPR} & ::= & \mathtt{+} \mid \mathtt{*} \mid \mathtt{<} \mid \mathtt{==} \\
\mathtt{PID} & ::= & \mathtt{p1} \mid \mathtt{...} \mid \mathtt{pk} \\
\mathtt{FID} & ::= & \mathtt{f0} \mid \mathtt{f1} \mid \mathtt{...} \mid \mathtt{fn} \\
\ulcorner\mathbb{B}\lrcorner & ::= & \mathtt{true} \mid \mathtt{false} \\
\ulcorner\mathbb{N}\lrcorner & ::= & \mathtt{1} \mid \mathtt{2} \mid \mathtt{3} \mid \mathtt{...}
\end{array}
$$

Type universe: Finite $\mathbb{T}$ with two primitive types (e.g., $\mathbb{B}, \mathbb{N}, \ldots$)

Ambient context: Γ maps $\mathtt{f\_} : (\tau_1, \ldots, \tau_m) \to \tau$.

Expression fragment: static semantics

$$\frac{\Gamma \vdash e_c : \mathbb{B} \qquad \Gamma \vdash e_\top : \tau \qquad \Gamma \vdash e_\perp : \tau}{\Gamma \vdash \texttt{if}\ e_c\ \texttt{\{}\ e_\top\ \texttt{\}}\ \texttt{else}\ \texttt{\{}\ e_\perp\ \texttt{\}} : \tau}\ \texttt{IFE}$$

$$\frac{\Gamma \vdash \texttt{f\_} : (\tau_1, \ldots, \tau_m) \to \tau \qquad \Gamma \vdash e_i : \tau_i \ \ \forall i \in [1, m]}{\Gamma \vdash \texttt{f\_}\ \texttt{(}\ e_1\ \texttt{,}\ \ldots\ \texttt{,}\ e_m\ \texttt{)} : \tau}\ \texttt{INV}$$

$$\frac{\delta_{\texttt{OPR}}(\odot, \tau, \tau') = \hat{\tau} \qquad \Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash \texttt{(}\ e_1\ \odot\ e_2\ \texttt{)} : \hat{\tau}}\ \texttt{OPX}$$

Where the operator typing function $\delta_{\texttt{OPR}} : \Sigma_{\texttt{OPR}} \times \mathbb{T} \times \mathbb{T} \rightharpoonup \mathbb{T}$ returns:

$$\delta_{\texttt{OPR}}(\odot, \tau, \tau') = \begin{cases} \mathbb{B} & \odot = \texttt{<},\ \tau = \tau' = \mathbb{B} \\ \mathbb{N} & \odot \in \{\texttt{+}, \texttt{*}\},\ \tau = \tau' = \mathbb{N} \\ \mathbb{B} & \odot = \texttt{==},\ \tau = \tau' \end{cases}$$

Embedding the type checker (I)

Grammar: $\langle \Sigma, V, P \subset V \times (V \cup \Sigma)^*, S \in V \rangle \Rightarrow \langle \Sigma_\Gamma, V_\Gamma, P_\Gamma, V_\Gamma, S_\Gamma \rangle$

Decorated nonterminals: $\mathtt{EXP}[\tau, \pi] \qquad \left(\tau \in \mathbb{T},\ \pi \equiv (\vec{\tau} \to \dot{\tau})\right)$

Provide: k , the maximum arity, and $\mathbb{T}$, the type universe.

$$\frac{\langle \vec{\tau}, \dot{\tau} \rangle \in \mathbb{T}^{0..k} \times \mathbb{T} \qquad \vec{\tau}_{0..|\vec{\tau}|} \in \vec{\tau}}{\left(S_\Gamma \to \mathtt{fun}\ \mathtt{f0}\ \mathtt{(}\ \mathop{\boldsymbol{,}}_{i=1}^{|\vec{\tau}|} \left(p_i\ \mathtt{:}\ \vec{\tau}_i\right)\ \mathtt{)}\ \mathtt{:}\ \dot{\tau}\ \mathtt{=}\ \mathtt{EXP}[\dot{\tau}, \vec{\tau} \to \dot{\tau}]\right) \in P_\Gamma}\ \mathtt{FUN}_\varphi$$

$$\frac{\mathtt{EXP}[\tau, \vec{\tau} \to \dot{\tau}] \in V_\Gamma \qquad \tau = \dot{\tau} \qquad \vec{\tau}_{0..|\vec{\tau}|} \in \vec{\tau}}{\left(\mathtt{EXP}[\tau, \vec{\tau} \to \dot{\tau}] \to \mathtt{f0}\ \mathtt{(}\ \mathop{\boldsymbol{,}}_{i=1}^{|\vec{\tau}|} \mathtt{EXP}[\vec{\tau}_i, \vec{\tau} \to \dot{\tau}]\ \mathtt{)}\right) \in P_\Gamma}\ \mathtt{REC}_\varphi$$

$$\frac{\mathtt{EXP}[\tau, \vec{\tau} \to \dot{\tau}] \in V_\Gamma \quad \exists i.\ \vec{\tau}_i = \tau}{\left(\mathtt{EXP}[\tau, \vec{\tau} \to \dot{\tau}] \to \mathtt{pi}\right) \in P_\Gamma}\ \mathtt{PID}_\varphi \qquad \frac{\mathtt{EXP}[\tau, \pi] \in V_\Gamma \quad \_ : \mathbb{B} \mid \mathbb{N}}{\left(\mathtt{EXP}[\tau, \pi] \to \_\right) \in P_\Gamma}\ \ulcorner \mathbb{T} \lrcorner_\varphi$$

Embedding the type checker (II)

$$\frac{\texttt{EXP}[\tau,\pi] \in V_\Gamma \qquad \Gamma \vdash \texttt{f\_} : (\tau_1,\ldots,\tau_m) \to \tau}{\left(\texttt{EXP}[\tau,\pi] \to \texttt{f\_}\ \texttt{(}\ \overset{m}{\underset{i=1}{\boldsymbol{,}}}\ \texttt{EXP}[\tau_i,\pi]\ \texttt{)}\right) \in P_\Gamma}\ \texttt{INV}_\varphi$$

$$\frac{\texttt{EXP}[\tau,\pi] \in V_\Gamma \qquad \tau = \tau' \qquad \tau,\tau' \in \mathbb{T}}{\left(\texttt{EXP}[\tau,\pi] \to \texttt{if}\ \texttt{EXP}[\mathbb{B},\pi]\ \texttt{\{}\ \texttt{EXP}[\tau,\pi]\ \texttt{\}}\ \texttt{else}\ \texttt{\{}\ \texttt{EXP}[\tau',\pi]\ \texttt{\}}\right) \in P_\Gamma}\ \texttt{IFE}_\varphi$$

$$\frac{\texttt{EXP}[\hat{\tau},\pi] \in V_\Gamma \qquad \delta_{\texttt{OPR}}(\odot,\tau,\tau') = \hat{\tau} \qquad \odot \in \{\texttt{==}, \texttt{<}, \texttt{+}, \texttt{*}\}}{\left(\texttt{EXP}[\hat{\tau},\pi] \to \texttt{(}\ \texttt{EXP}[\tau,\pi]\ \odot\ \texttt{EXP}[\tau',\pi]\ \texttt{)}\right) \in P_\Gamma}\ \texttt{OPX}_\varphi$$

Finally, we normalize to Chomsky Normal Form (CNF), rewriting all productions to either **(1)** $(w \to xz) : V \times V^2$ or **(2)** $(w \to t) : V \times \Sigma$.

Addendum: CFG $\cap$ NFA closure and $G_\cap$ construction

Bar-Hillel (1961): For any CFG G, and NFA $A = \langle Q, \Sigma, \delta, q_\alpha, F\rangle$, $\exists G_\cap$ s.t. $\mathcal{L}(G_\cap) = \mathcal{L}(G) \cap \mathcal{L}(A)$. Salomaa's (1973) construction:

$$\frac{q_\omega \in F}{\big(S_\cap \to q_\alpha\, S\, q_\omega\big) \in P_\cap}\,\mathcal{S} \qquad \frac{(W \to a) \in P \qquad (p \xrightarrow{a} r) \in \delta}{\big(pWr \to a\big) \in P_\cap}\uparrow$$

$$\frac{(W \to XZ) \in P \qquad p, q, r \in Q}{\big(pWr \to (pXq)\,(qZr)\big) \in P_\cap}\bowtie$$

but, there is a *much* more efficient construction. Intuition: want to show $q_\alpha \rightsquigarrow q_\omega$ in A such that $q_\omega : F$ where $q_\alpha \rightsquigarrow q_\omega \vdash S$. At least one of two cases must hold for $w \in V$ to parse a given $p \rightsquigarrow r$ pair:

1. $\exists a.\big((p \xrightarrow{a} r) \in \delta \land (w \to a) \in P\big)$, or,
2. $\exists q, x, z.\Big((w \to xz) \in P \land \overbrace{\underbrace{p \rightsquigarrow q}_{x}, \underbrace{q \rightsquigarrow r}_{z}}^{w}\Big)$.

Finite intersection as matrix exponentiation on $(2^V, \oplus, \otimes)$

Let $M \in (2^V)^{|Q|\times|Q|}$, with entries $M[r,c] \subseteq V$ (a set of nonterminals), and let $X \oplus Z = X \cup Z, X \otimes Z = \left\{ w \,\middle|\, \exists x \in X,\ z \in Z.\, (w \to xz) \in P \right\}$.

$$M_0[r,c] \;=\; \bigcup_{a \in \Sigma} \left\{ w \,\middle|\, (w \to a) \in P \;\land\; (q_r \overset{a}{\to} q_c) \in \delta \right\}.$$

We will define the matrix exponential in the standard manner:

$$\exp(M_0) = \sum_{i=0}^{\infty} M_0^i \;=\; \sum_{i=0}^{|Q|} M_0^i \quad (\alpha_\varnothing \Leftrightarrow \text{S.U.T.} \Rightarrow \text{nilpotent}).$$

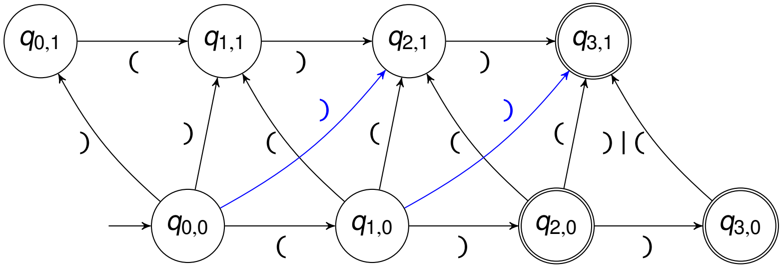
$$T(2n) = \sum_{i=0}^{2n} M_0^i = \begin{cases} M_0, & n = 1, \\ T(n) \;\oplus\; \big(T(n) \cdot T(n)\big), & \text{otherwise.} \end{cases}$$

The following proposition decides nonemptiness:

$$\left[\; \bigvee_{q_\omega \in F} S \in \exp(M_0)[q_\alpha, q_\omega] \;\right] \Longleftrightarrow \mathcal{L}(G) \cap \mathcal{L}(\alpha_\cap) \neq \varnothing$$

Repair example: Simple Levenshtein automaton

Suppose we have the string, $\sigma = \texttt{(}\ \texttt{)}\ \texttt{)}$ and wish to balance the parentheses. Assume we have the Chomsky Normal Form CFG, $G' = \left\{S \to LR, S \to LF, S \to SS, F \to SR, L \to (, R \to)\right\}$ and let us impose an ordering of S, F, L, R on V. We will initially have the Levenshtein automaton, $\alpha_{\varnothing}$, depicted below:



n.b. acyclic, therefor has strictly upper triangular adjacency matrix.

Repair example: Initial parse chart (M_0)

M_0	q_{00}	q_{01}	q_{10}	q_{11}	q_{20}	q_{21}	q_{30}	q_{31}
		S F L R	*S F L R*	*S F L R*	*S F L R*	*S F L R*	*S F L R*	*S F L R*
q_{00}		□□□■	□□■□	□□□■	□□□□	□□□■	□□□□	□□□□
q_{01}			□□□□	□□■□	□□□□	□□□□	□□□□	□□□□
q_{10}				□□■□	□□□■	□□■□	□□□□	□□□■
q_{11}					□□□□	□□□■	□□□□	□□□□
q_{20}						□□■□	□□□■	□□■□
q_{21}							□□□□	□□□■
q_{30}								□□■■
q_{31}								

Initial configuration, after filling all unit productions.

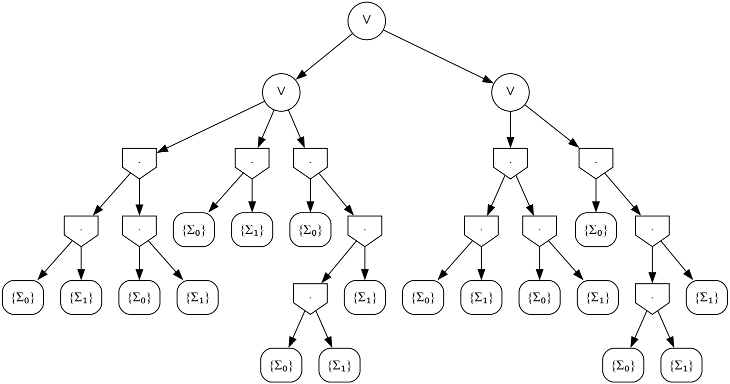
Repair example: Final parse chart (e^{M_0})

M_∞	q_{00}	q_{01}	q_{10}	q_{11}	q_{20}	q_{21}	q_{30}	q_{31}
		$S\,F\,L\,R$	$S\,F\,L\,R$	$S\,F\,L\,R$	$S\,F\,L\,R$	$S\,F\,L\,R$	$S\,F\,L\,R$	$S\,F\,L\,R$
q_{00}		□□□■	□□■□	□□□■	■□□□	□□□■	□■□□	■□□□
q_{01}			□□□□	□□■□	□□□□	■□□□	□□□□	□■□□
q_{10}				□□■□	□□□■	■□■□	□□□□	■■□■
q_{11}					□□□□	□□□■	□□□□	□□□□
q_{20}						□□■□	□□□■	■□■□
q_{21}							□□□□	□□□■
q_{30}								□□■■
q_{31}								

Final configuration, after matrix fixpoint is reached.

Repair example: Regex denoting $\mathcal{L}(G) \cap \mathcal{L}(\alpha_\varnothing)$

$$\Big(\texttt{a b a b} \mid \big(\texttt{a b} \mid \texttt{a a b b}\big)\Big) \Big| \big(\texttt{a b a b} \mid \texttt{a a b b}\big)$$



Regular expression reconstructed from the final parse chart.

Sampling star-free regular expressions uniformly

Let $e : E$ be an SFRE with two connectives: $e \to \Sigma \mid e \cdot e \mid e \vee e$.

Theorem (Uniform tree enumeration)

To sample parse trees, take a PRNG and feed it into `enum`*:*

$$\mathtt{enum}(e, n) = \begin{cases} e & \text{if } e \in \Sigma \\ \mathtt{enum}\left(x, \lfloor \frac{n}{|z|} \rfloor\right) \cdot \mathtt{enum}\left(z,\ n \bmod |z|\right) & \text{if } e = x \cdot z \\ \mathtt{enum}\left((x, z)_{\min(1, \lfloor \frac{n}{|x|} \rfloor)}, n - |x| \min(1, \lfloor \frac{n}{|x|} \rfloor)\right) & \text{if } e = x \vee z \end{cases}$$

Where the number of parse trees in a SFRE we abbreviate as $|e|$:

$$|e| : E \to \mathbb{N} = \begin{cases} 1 & \text{if } e \in \Sigma \\ x \times z & \text{if } e = x \cdot z \\ x + z & \text{if } e = x \vee z \end{cases}$$

n.b. we may need to disambiguate to guarantee $\mathcal{L}(e)$ uniformity.

Sampling star-free regular expressions autoregressively

Now, for any SFRE, e, $\texttt{choose}(e)$ witnesses $\sigma \in \mathcal{L}(e)$:

$$\texttt{follow}(e) = \begin{cases} \{e\} & \text{if } e \in \Sigma \\ \texttt{follow}(x) & \text{if } e = x \cdot z \\ \texttt{follow}(x) \cup \texttt{follow}(z) & \text{if } e = x \vee z \end{cases}$$

$$\texttt{choose}(e) = \begin{cases} e & \text{if } e \in \Sigma \\ \big(s \leftsquigarrow \texttt{follow}(e)\big) \cdot \texttt{choose}(\partial_s e) & \text{if } e = x \cdot z \\ \texttt{choose}\big(e' \leftsquigarrow \{x, z\}\big) & \text{if } e = x \vee z \end{cases}$$

where $\delta_s e$ is the Brzozowskian derivative (1973) and $\leftsquigarrow$ denotes probabilistic choice from a small finite set. This may be augmented with a weighted choice operator, $\sigma \leftsquigarrow P_\theta(\sigma_n \mid \sigma_{n-1}, \cdots, \sigma_{n-k})$.

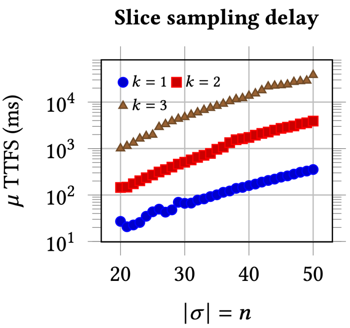
Evaluation benchmarks

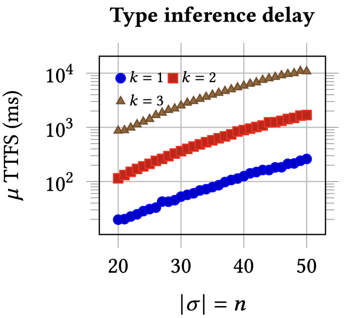
Experimental Setup

- Arity: $k \in \{1, 2, 3\}$
  Fixed: $|\Gamma| = 18$, $|\mathbb{T}| = 7$
- CNF grammar sizes:
  $|G'_\Gamma| \in [1.9 \times 10^4,\ 9.9 \times 10^5]$
- Apple M4 (16 GB RAM)

Benchmarks

- **Slicing**: $\sigma \twoheadleftarrow \mathcal{L}(G'_\Gamma) \cap \Sigma^n$
- **Type inference**: reuse random functions from slice sampling, replace $(\boxed{:}\ \tau\ \boxed{=})$ with $(\boxed{:}\ \Sigma\ \boxed{=})$, and $\sigma' \twoheadleftarrow \mathcal{L}(G'_\Gamma) \cap (\ldots \boxed{:}\ \Sigma\ \boxed{=} \ldots)$
- **Bounded delay**: 1786 ± 817 ns
- **Throughput**: $\sim 2.2 \times 10^7$ tok/s





Future work

- More compact embeddings and asymptotic complexity
- Laziness: instantiate CFG productions during parsing
- Extend to richer type systems, e.g., polymorphism, higher-order functions, subtyping, nested scope
- "A Relevance Sampler for $\mu\mathrm{Rust}_{\mathrm{sl}}$" (Considine, 2025) explores a straight-line fragment of Rust with nominal relevance
- "A Tree Sampler for Bounded CFLs" (Considine, 2024) describes a uniform sampler for finite CFL intersections
- Other FPT embeddings. Open to suggestions!
- Applications to program synthesis and repair
- Try it yourself at: https://tidyparse.github.io

Acknowledgements

Thank you!

- Ori Roth
- Chuta Sano
- Brigitte Pientka
- David Bieber
- Margaret Considine
- Mark Considine

Lastly, thank you to the LAFI organizers!