A Word Sampler for Well-Typed Functions

Breandan Mark Considine

January 11, 2026

Formal langauges & type theory

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

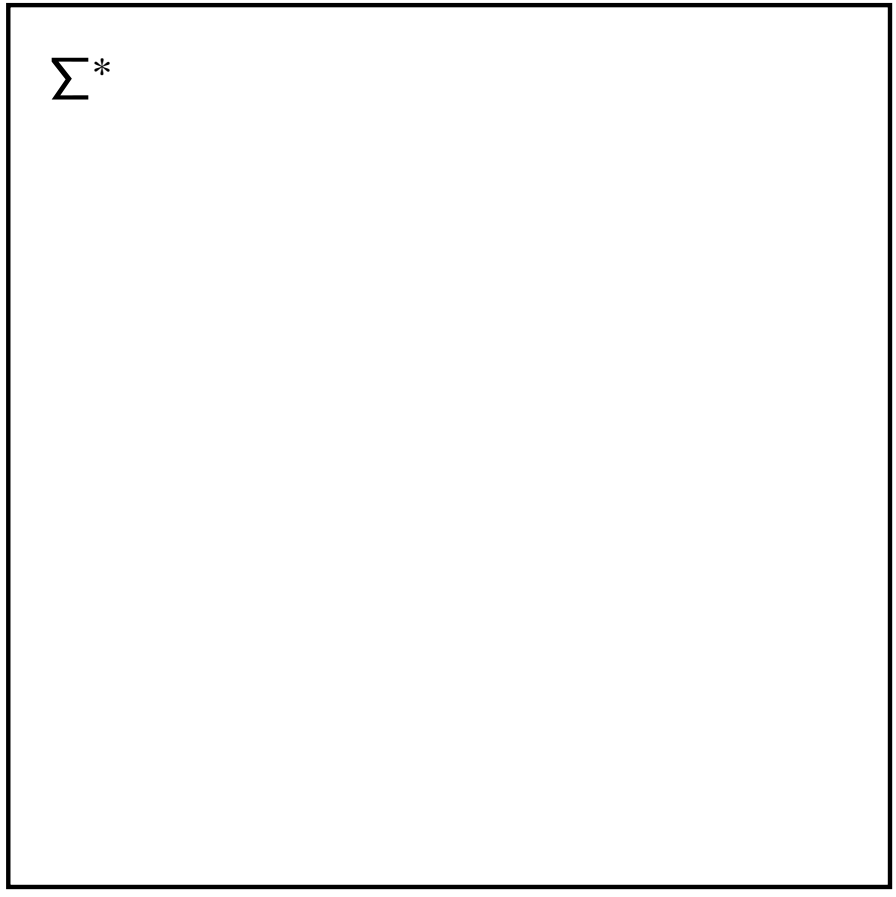
$$\underbrace{(W \to XZ) \in P}_{\text{grammar production}} \quad \leftrightsquigarrow \quad \underbrace{\frac{\Gamma \vdash x : X \qquad \Gamma \vdash z : Z}{\Gamma \vdash xz : W}}_{\text{typing judgment}}$$

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

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

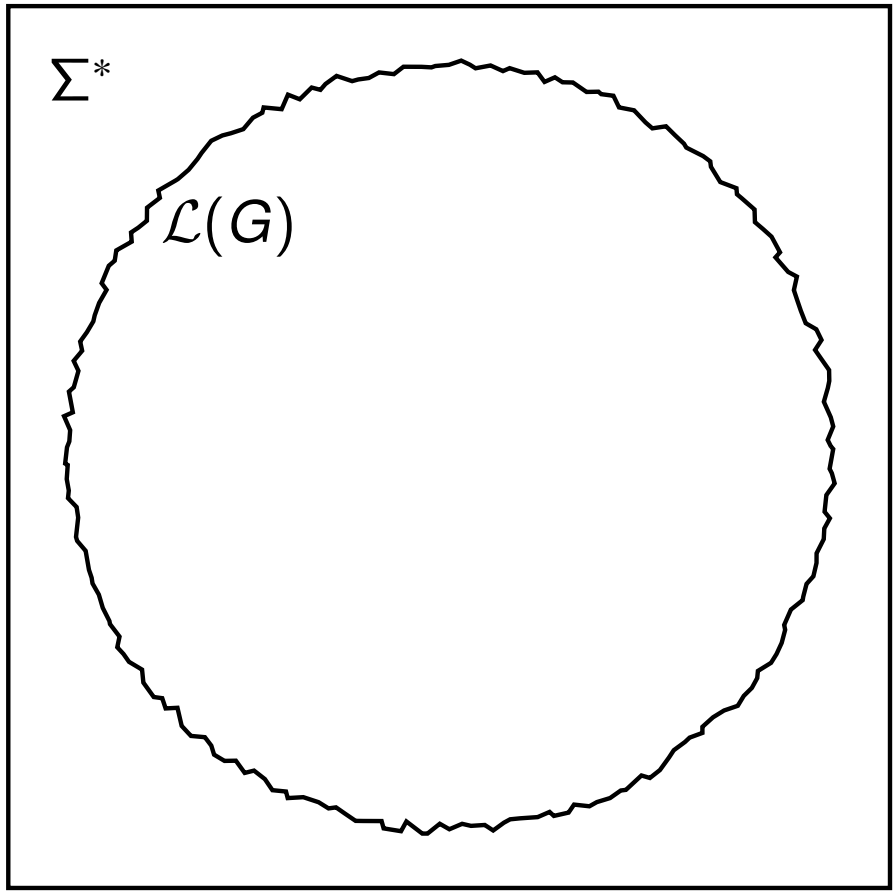
Programming language [in]approximability

- Σ^*: all words over Σ



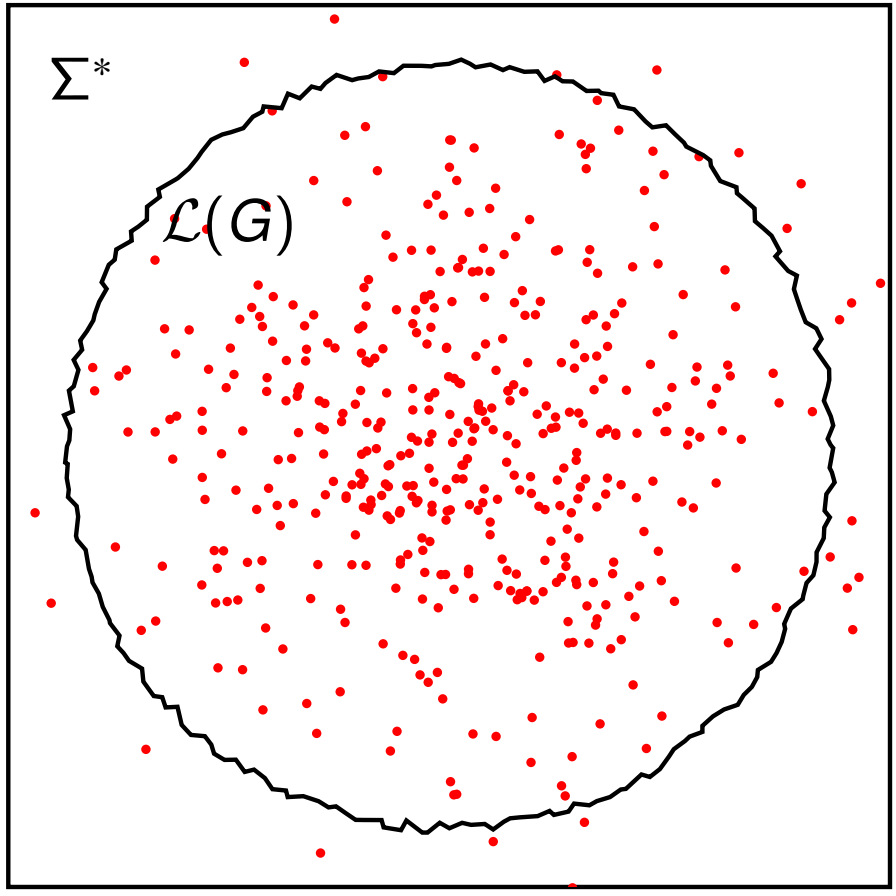
Programming language [in]approximability

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



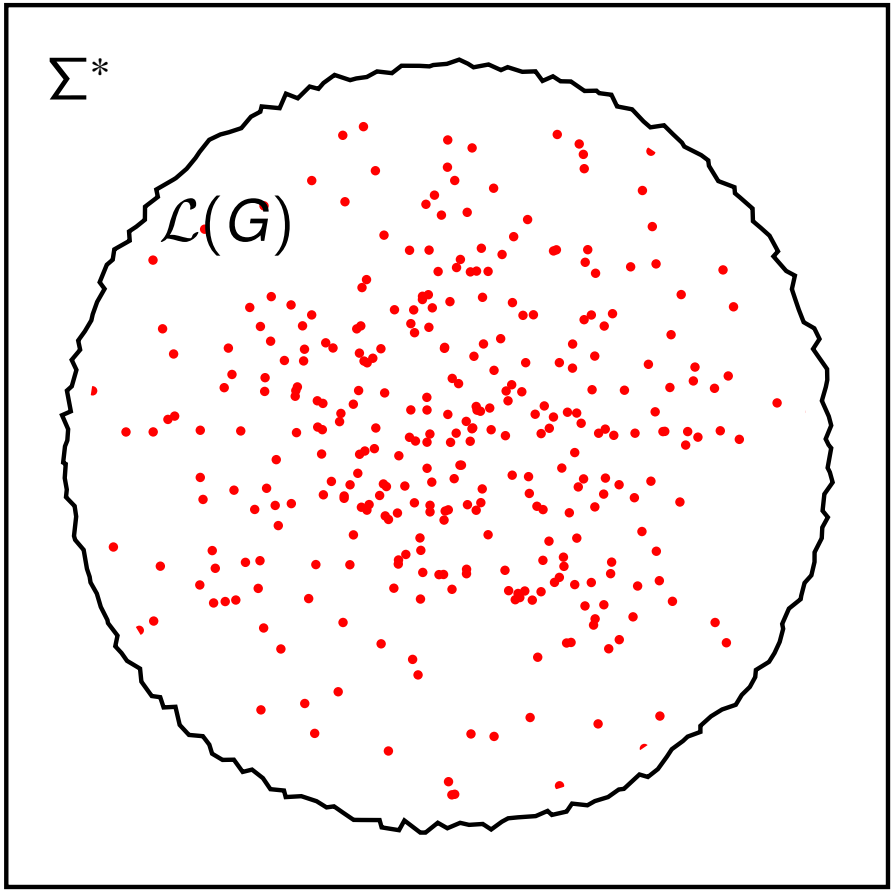
Programming language [in]approximability

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



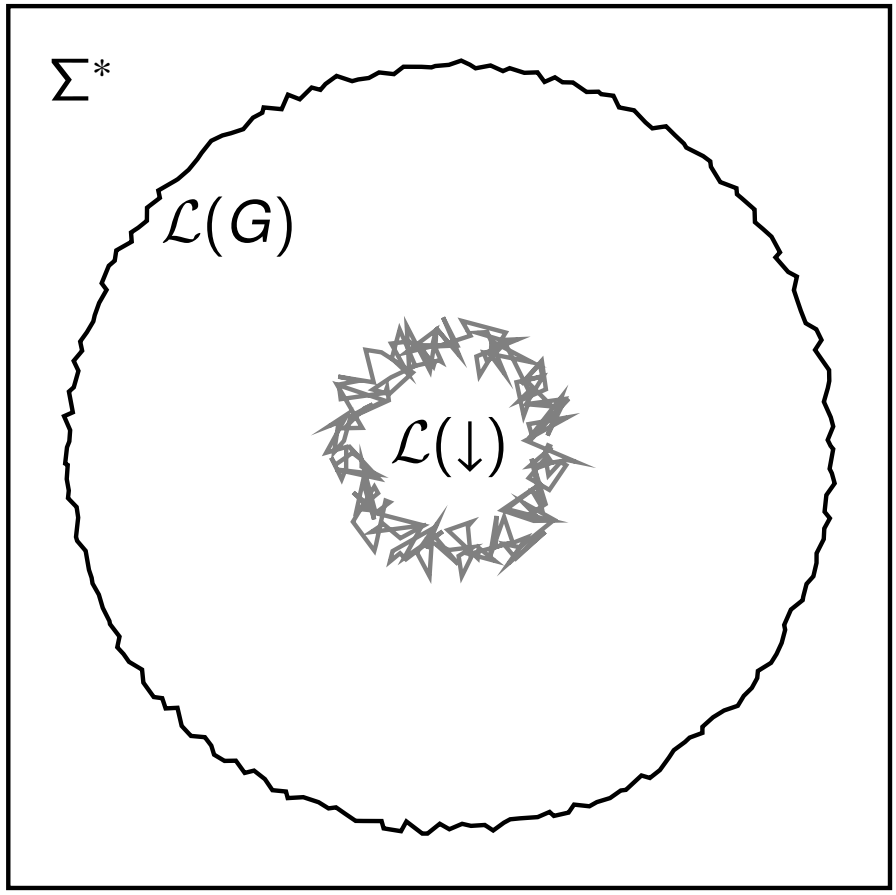
Programming language [in]approximability

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



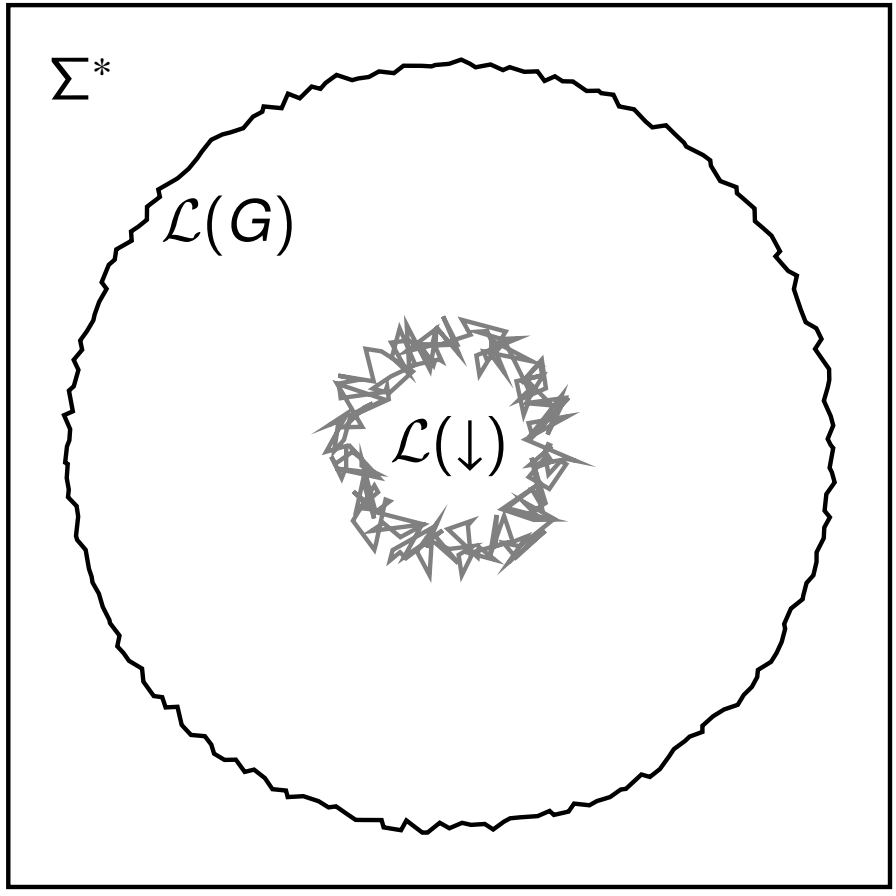
Programming language [in]approximability

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



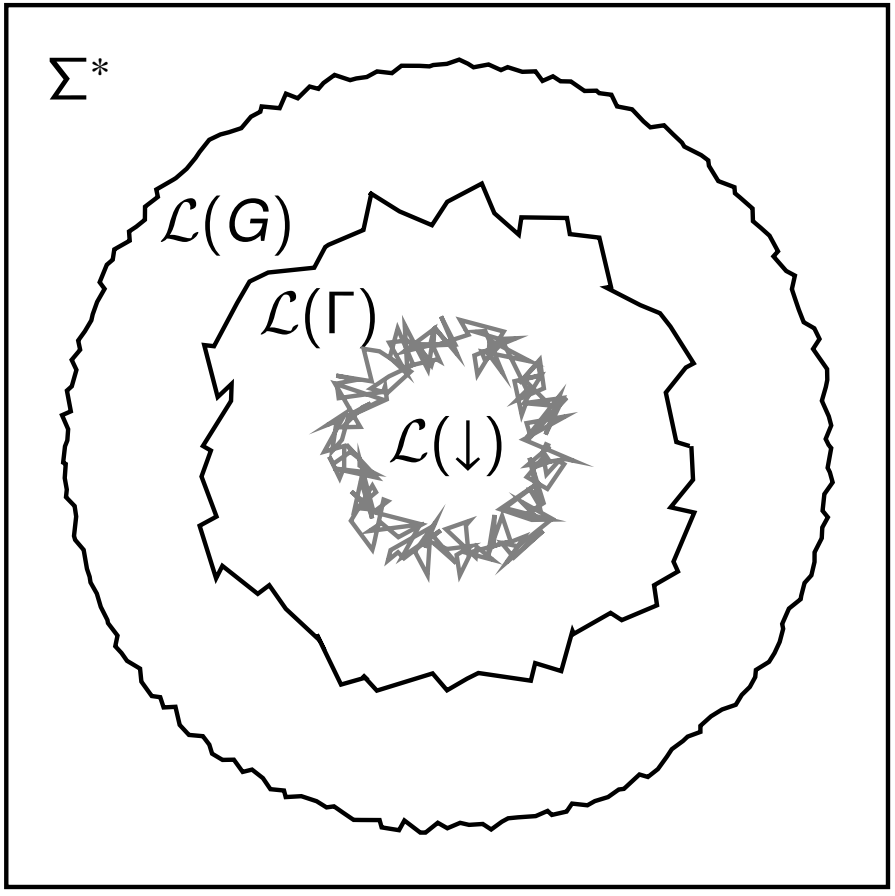
Programming language [in]approximability

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



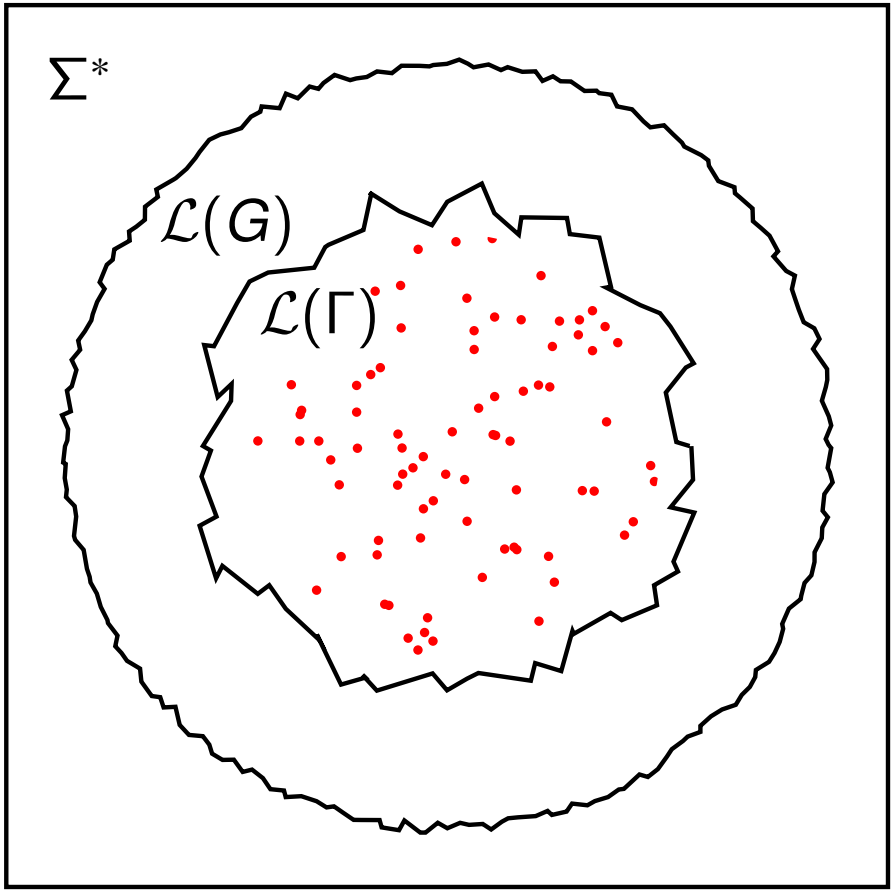
Programming language [in]approximability

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



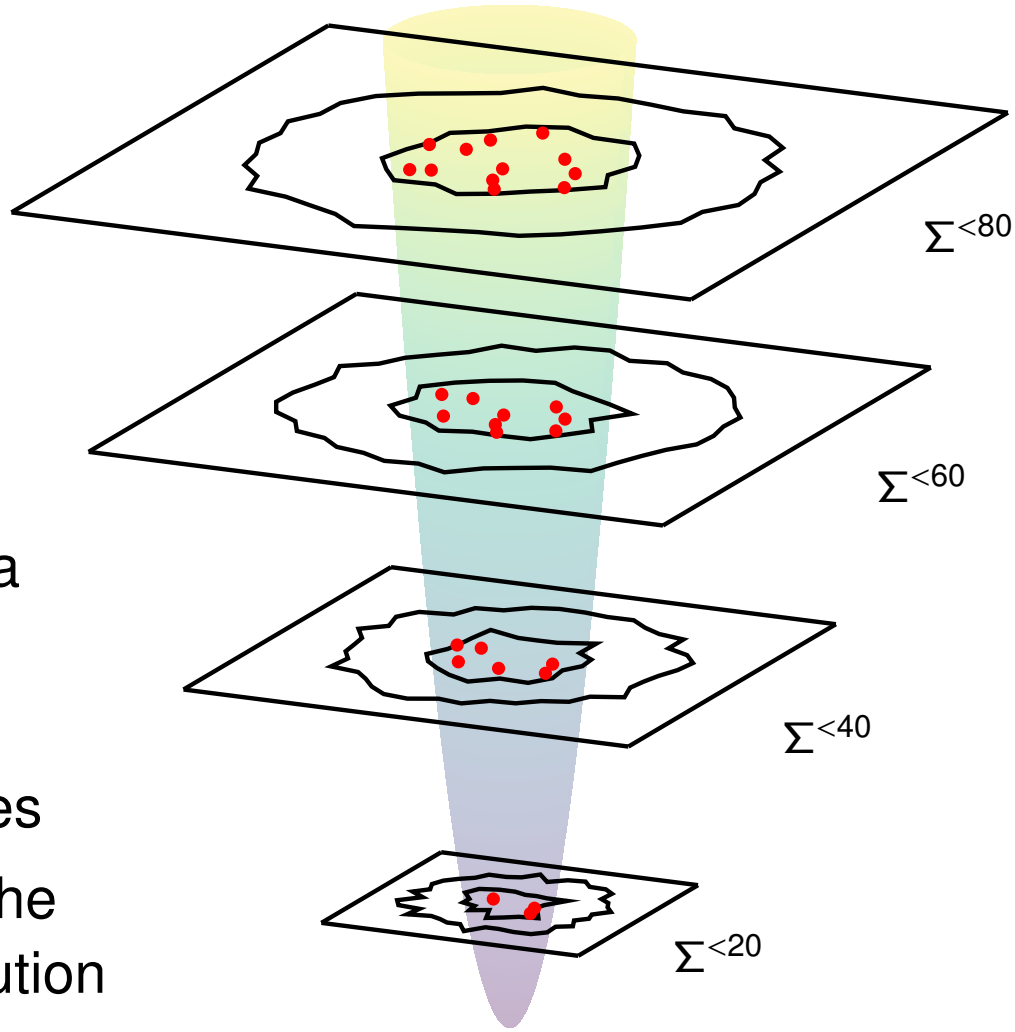
Programming language [in]approximability

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



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

```
FUN  ::= fun f0 ( PRM ) : 𝕋 = EXP
PRM  ::= PID : 𝕋 | PRM , PID : 𝕋
EXP  ::= ⌜ℕ⌟ | ⌜𝔹⌟ | PID | INV | IFE | OPX
OPX  ::= ( EXP OPR EXP )
IFE  ::= if EXP { EXP } else { EXP }
INV  ::= FID ( ARG )
ARG  ::= EXP | ARG , EXP
OPR  ::= + | * | < | ==
PID  ::= p1 | ... | pk
FID  ::= f0 | f1 | ... | fn
⌜𝔹⌟  ::= true | false
⌜ℕ⌟  ::= 1 | 2 | 3 | ...
```

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

Ambient context: Γ maps `f_` : $(\tau_1, \ldots, \tau_m) \to \tau$.

Expression fragment: static semantics

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

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

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

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

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

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

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

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

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

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

Embedding the type checker (II)

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

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

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

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

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 $X \oplus Z = X \cup Z, X \otimes Z = \left\{ w \,\middle|\, \exists x \in X,\ z \in Z.\ (w \to xz) \in P \right\}$.

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

We will define the matrix exponential in the standard manner:

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

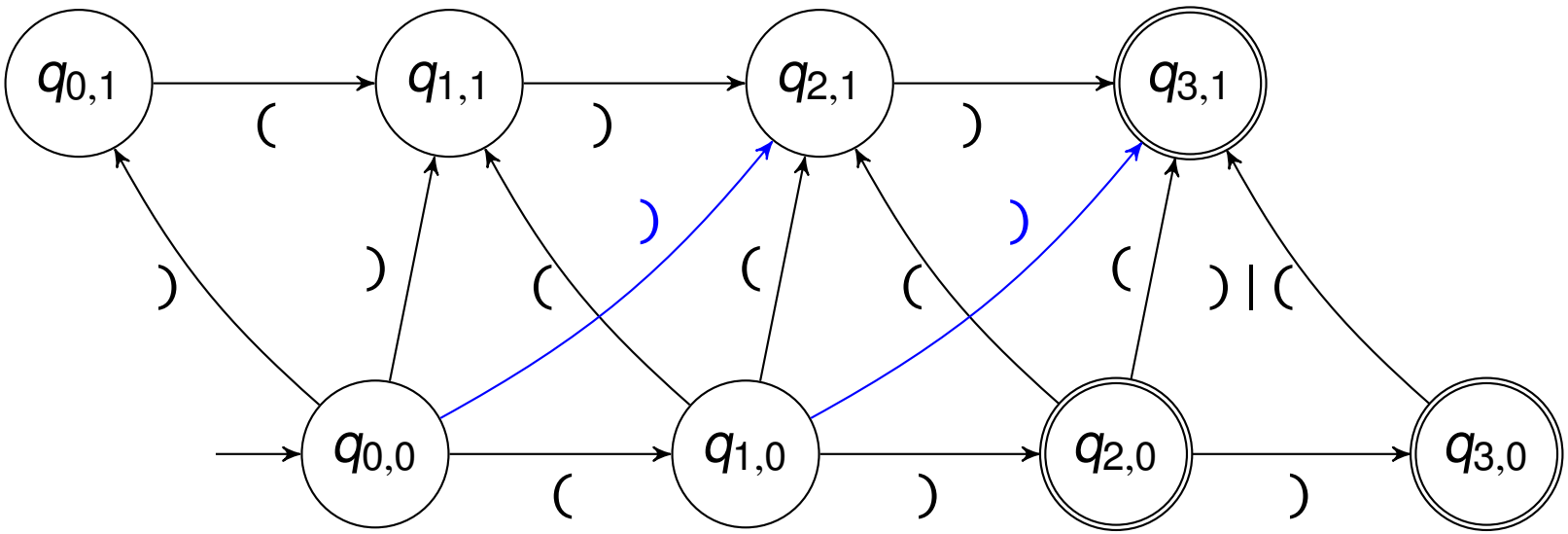
$$T(2n) = \sum_{i=0}^{2n} M_0^i = \begin{cases} M_0, & n = 1, \\ T(n) \ \oplus \ \Big(T(n) \cdot T(n) \Big), & \text{otherwise.} \end{cases}$$

The following proposition decides nonemptiness:

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

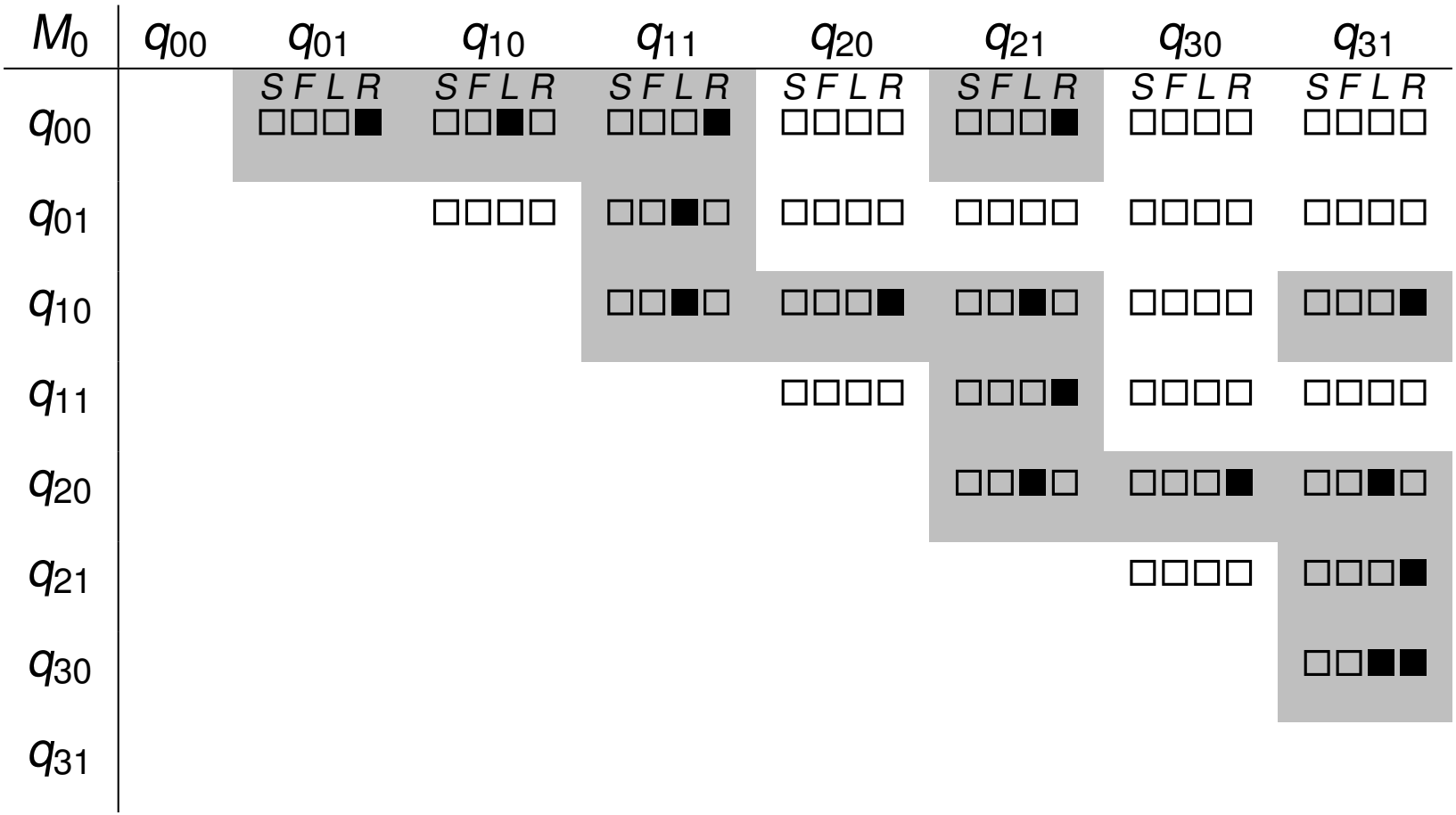
Repair example: Simple Levenshtein automaton

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



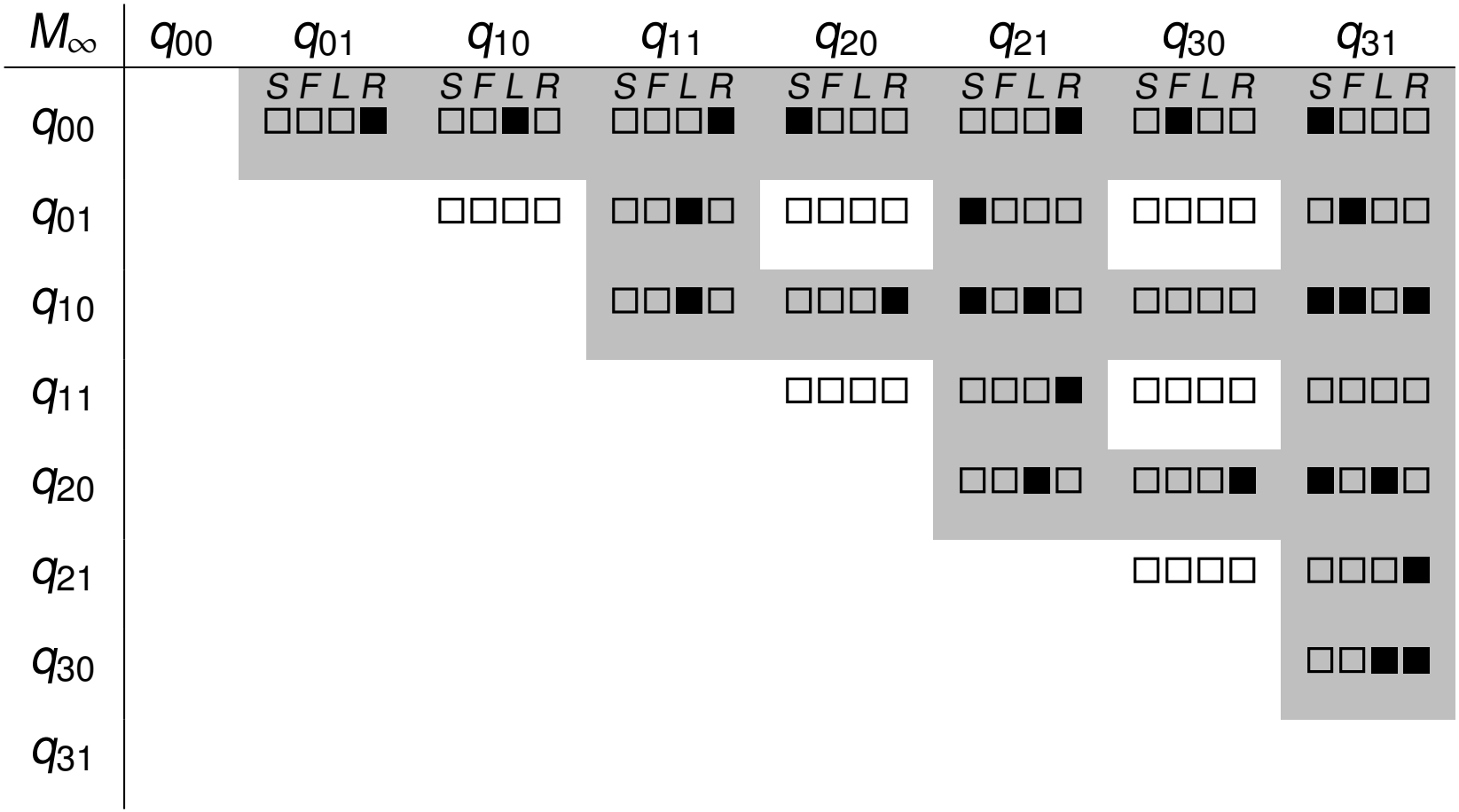
n.b. acyclic, therefor has strictly upper triangular adjacency matrix.

Repair example: Initial parse chart (M_0)



Initial configuration, after filling all unit productions.

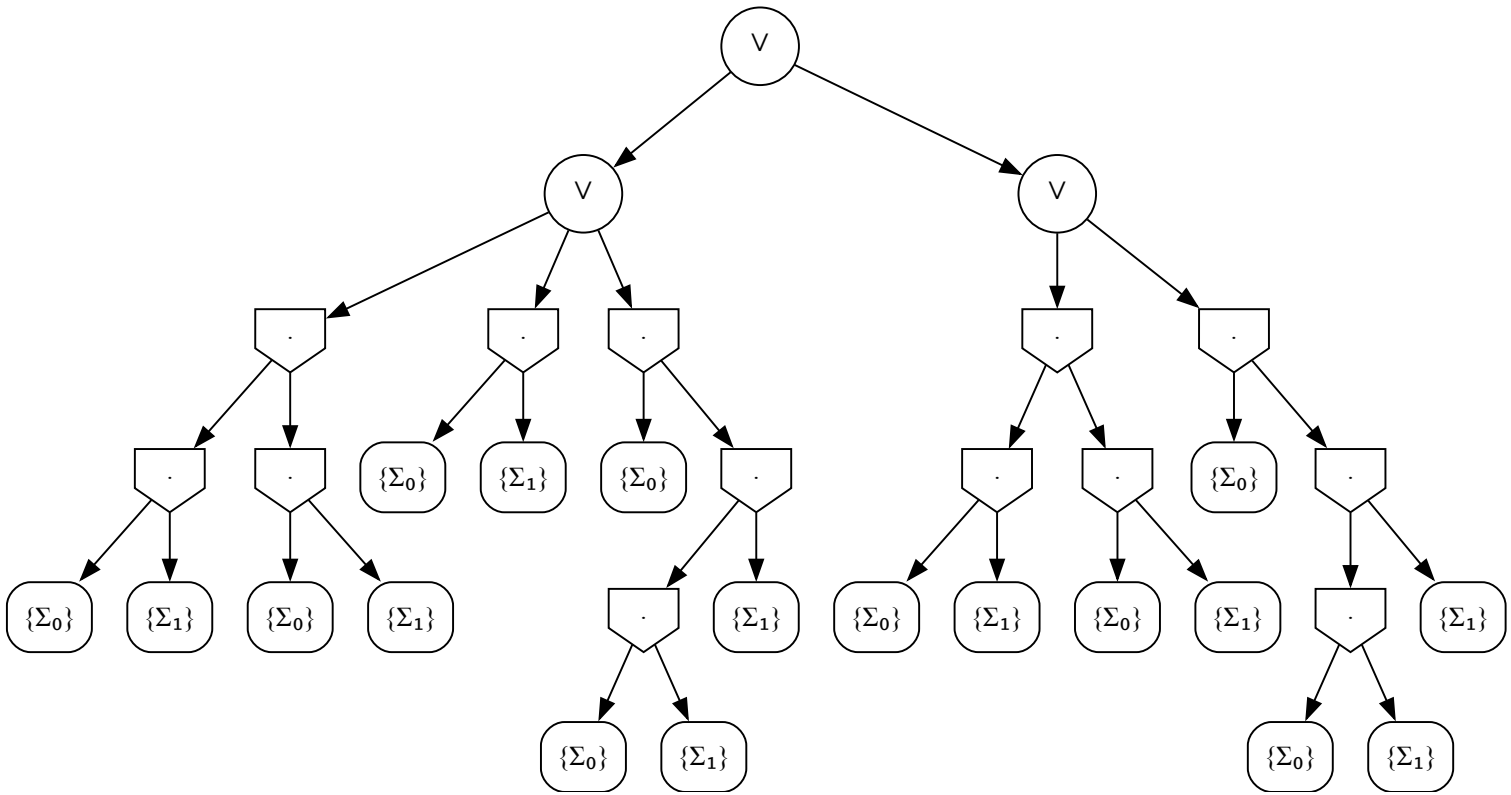
Repair example: Final parse chart (e^{M_0})



Final configuration, after matrix fixpoint is reached.

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

$$\Big(\mathtt{a}\ \mathtt{b}\ \mathtt{a}\ \mathtt{b} \mid \big(\mathtt{a}\ \mathtt{b} \mid \mathtt{a}\ \mathtt{a}\ \mathtt{b}\ \mathtt{b}\big)\Big) \mid \big(\mathtt{a}\ \mathtt{b}\ \mathtt{a}\ \mathtt{b} \mid \mathtt{a}\ \mathtt{a}\ \mathtt{b}\ \mathtt{b}\big)$$



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 $\mathtt{enum}$*:*

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

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, $\sigma \leftsquigarrow P_\theta\,(\sigma_n \mid \sigma_{n-1}, \cdots, \sigma_{n-k})$.

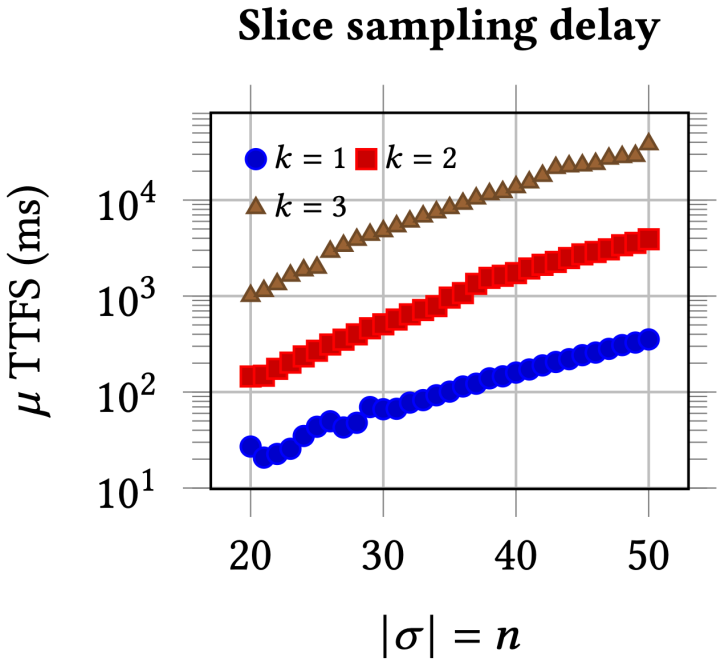
Evaluation benchmarks

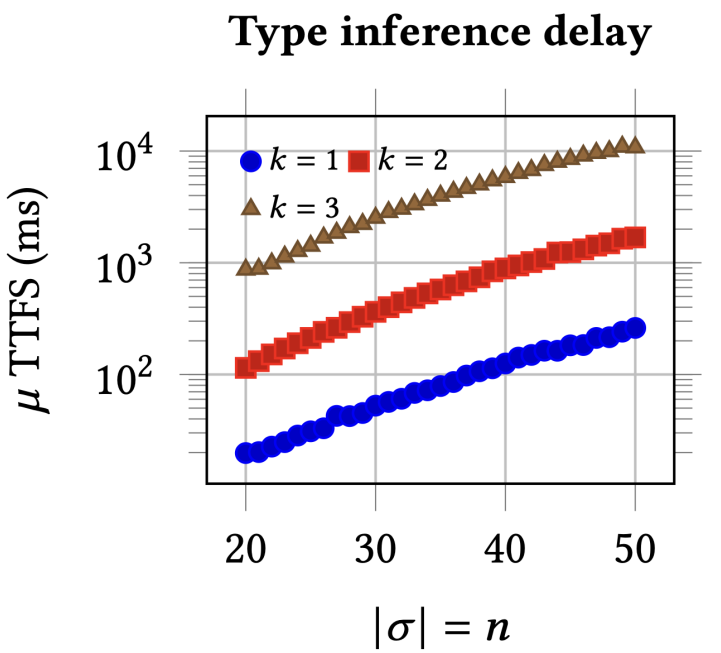
Experimental Setup

- 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**: $\sigma \leftsquigarrow \mathcal{L}(G'_\Gamma) \cap \Sigma^n$
- **Type inference**: reuse random functions from slice sampling, replace $(\texttt{:}\ \tau\ =)$ with $(\texttt{:}\ \Sigma\ =)$, and $\sigma' \leftsquigarrow \mathcal{L}(G'_\Gamma) \cap (\ldots \texttt{:}\ \Sigma\ = \ldots)$
- **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 $\mu\text{Rust}_{\text{sl}}$" (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!