Probabilistic Array Programming on Galois Fields

Breandan Considine, Jin Guo, Xujie Si

Main Idea

- Boolean matrices are useful structures for simulating finite state machines
- The operators $\{\text{XOR}, \land, \top\}$ are *functionally complete* logical connectives
- We implement sketch-based probabilistic context-free program synthesis

Algebraic Parsing

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

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

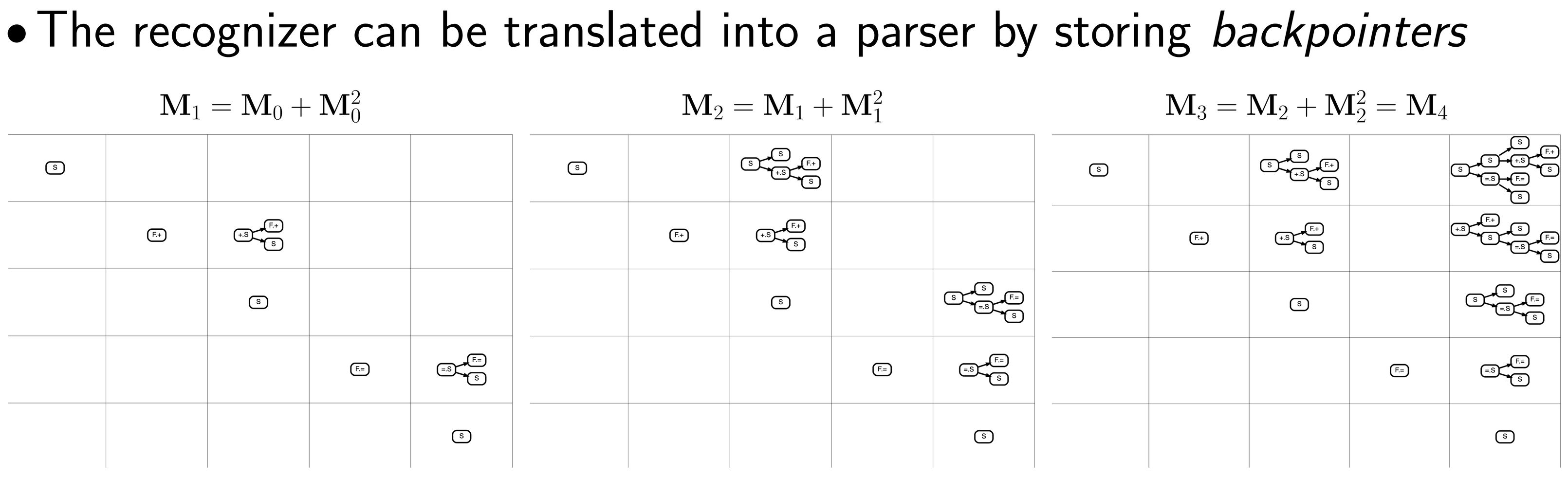
Initializing $\mathbf{M}_0[i, j](\mathcal{G}, \sigma) \coloneqq \{A \mid i+1 = j, (A \to \sigma_i) \in P\}$ and searching for the least solution to $\mathbf{M} = \mathbf{M} + \mathbf{M}^2$, will produce a fixedpoint $\mathbf{M}^*$:

$$\mathbf{M}^* = \begin{pmatrix} \varnothing & \{V\}_{\sigma_1} & \ldots & \ldots & \mathcal{T} \\ \varnothing & \varnothing & \{V\}_{\sigma_2} & \ldots & \ldots \\ \varnothing & \varnothing & \varnothing & \{V\}_{\sigma_3} & \ldots \\ \varnothing & \varnothing & \varnothing & \varnothing & \{V\}_{\sigma_4} \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \end{pmatrix}$$

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

Parsing Dynamics

- The matrix $\mathbf{M}_0$ is strictly upper triangular, i.e., nilpotent of degree n
- The recognizer can be translated into a parser by storing *backpointers*



- If we had a way to solve for $\mathbf{M} = \mathbf{M} + \mathbf{M}^2$ directly, power iteration would be unnecessary and we could solve for $\mathbf{M} = \mathbf{M}^2$ above the superdiagonal...

Binarized CFL Sketching

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

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

- These operators can be lifted into matrices and tensors in the usual way
- In most cases, only a few nonterminals will be active at any given time
- More sophisticated representations are known for $\binom{n}{0\leq k}$ subsets

Feedback Shift Registers

Let $\mathbf{M} : \text{GF}(2^{n\times n})$ be a square matrix $\mathbf{M}^0_{r,c} = P_c$ if $r = 0$ else $\mathbb{1}[c = r - 1]$, where P is a feedback polynomial with coefficients $P_{1\ldots n}$ and $\oplus \coloneqq \veebar, \otimes \coloneqq \land$:

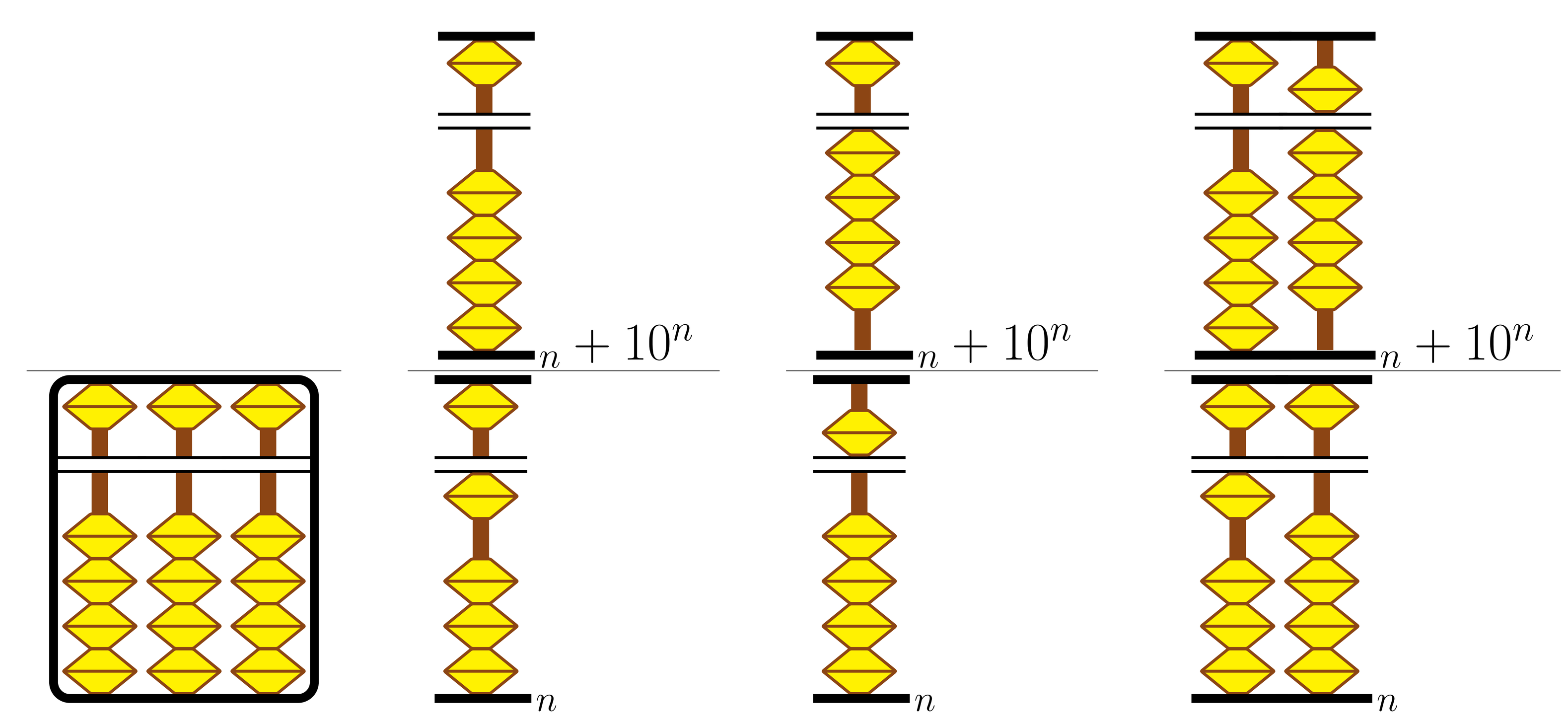
$$\mathbf{M}^t V = \begin{pmatrix} P_1 & P_2 & P_3 & P_4 & P_5 \\ \top & \circ & \circ & \circ & \circ \\ \circ & \top & \circ & \circ & \circ \\ \circ & \circ & \top & \circ & \circ \\ \circ & \circ & \circ & \top & \circ \end{pmatrix}^t \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{pmatrix}$$

Selecting any $V \neq \mathbf{0}$ and coefficients P_j from a known *primitive polynomial*, then powering the matrix $\mathbf{M}$ generates an ergodic sequence over $\text{GF}(2^n)$:

$$\mathbf{S} = \begin{pmatrix} V & \mathbf{M}V & \mathbf{M}^2V & \mathbf{M}^3V & \cdots & \mathbf{M}^{2^n-1}V \end{pmatrix}$$

This sequence has *full periodicity*, i.e., for all $i, j \in [0, 2^n), \mathbf{S}_i = \mathbf{S}_j \Rightarrow i = j$.

Typelevel Modular Arithmetic




```
// Typelevel Church encoding of dependently typed binary arithmetic
val t: T<T<T<T<F<T<0>>>>>> = T.F.T * T.F.F.T + T.F.T - T.T.F / T.F

// Typelevel Fibonacci configuration linear feedback shift register
val lfsr5 = BVec(T, F, F, T, T)
   .lfsr().lfsr().lfsr().lfsr().lfsr().lfsr() // BVec5<_, _, T, T, T>
   .lfsr().lfsr().lfsr().lfsr().lfsr().lfsr() // BVec5<T, T, _, T, T>
   .lfsr().lfsr().lfsr().lfsr().lfsr().lfsr() // BVec5<_, T, _, T, _>

// Typelevel implementation of Rule 110 elementary cellular automaton
val eca10 = BVec(T, T, F, F, F, T, F, F, F, F)
   .eca(::r110, ::r110, ...)  // BVec10<T, T, _, _, T, T, _, _, _, T>
   .eca(::r110, ::r110, ...)  // BVec10<T, T, _, T, T, T, _, _, T, T>
   .eca(::r110, ::r110, ...)  // BVec10<T, T, T, T, _, T, _, T, T, T>
   .eca(::r110, ::r110, ...)  // BVec10<_, _, _, T, T, T, T, T, _, _>
```

Tidyparse IDE Plugin