The Hilbert method for homomorphism equivalence in indexed languages

Richard Mandel

Queen Mary University of London

The homomorphism equivalence problem

Fix a class of languages $\mathcal{C}$. For a language $\Sigma^* \supseteq L \in \mathcal{C}$ and homomorphisms h, k of Σ^*, the *homomorphism equivalence problem for $\mathcal{C}$*, or **HEP**$(\mathcal{C})$, asks whether $h|_L = k|_L$.

- `INPUT:` $L \in \mathcal{C}$ (e.g. given by a grammar), $h, k : \Sigma^* \to \Sigma^*$
- `OUTPUT:` "YES" if $h|_L = k|_L$; "NO" otherwise.

The homomorphism equivalence problem

Fix a class of languages $\mathcal{C}$. For a language $\Sigma^* \supseteq L \in \mathcal{C}$ and homomorphisms h, k of Σ^*, the *homomorphism equivalence problem for $\mathcal{C}$*, or **HEP**($\mathcal{C}$), asks whether $h|_L = k|_L$.

- `INPUT:` $L \in \mathcal{C}$ (e.g. given by a grammar), $h, k : \Sigma^* \to \Sigma^*$
- `OUTPUT:` "YES" if $h|_L = k|_L$; "NO" otherwise.

- Studied by Culik, Salomaa et al. in the 70s

The homomorphism equivalence problem

Fix a class of languages $\mathcal{C}$. For a language $\Sigma^* \supseteq L \in \mathcal{C}$ and homomorphisms h, k of Σ^*, the *homomorphism equivalence problem for $\mathcal{C}$*, or **HEP**($\mathcal{C}$), asks whether $h|_L = k|_L$.

- `INPUT:` $L \in \mathcal{C}$ (e.g. given by a grammar), $h, k : \Sigma^* \to \Sigma^*$
- `OUTPUT:` "YES" if $h|_L = k|_L$; "NO" otherwise.

- Studied by Culik, Salomaa et al. in the 70s
- Decidability shown for:

The homomorphism equivalence problem

Fix a class of languages $\mathcal{C}$. For a language $\Sigma^* \supseteq L \in \mathcal{C}$ and homomorphisms h, k of Σ^*, the *homomorphism equivalence problem for $\mathcal{C}$*, or **HEP**($\mathcal{C}$), asks whether $h|_L = k|_L$.

- `INPUT:` $L \in \mathcal{C}$ (e.g. given by a grammar), $h, k : \Sigma^* \to \Sigma^*$
- `OUTPUT:` "YES" if $h|_L = k|_L$; "NO" otherwise.

- Studied by Culik, Salomaa et al. in the 70s
- Decidability shown for:
  1. Context free languages (Culik-Salomaa '78)

The homomorphism equivalence problem

Fix a class of languages $\mathcal{C}$. For a language $\Sigma^* \supseteq L \in \mathcal{C}$ and homomorphisms h, k of Σ^*, the *homomorphism equivalence problem for $\mathcal{C}$*, or **HEP**($\mathcal{C}$), asks whether $h|_L = k|_L$.

- `INPUT:` $L \in \mathcal{C}$ (e.g. given by a grammar), $h, k : \Sigma^* \to \Sigma^*$
- `OUTPUT:` "YES" if $h|_L = k|_L$; "NO" otherwise.

- Studied by Culik, Salomaa et al. in the 70s
- Decidability shown for:
  1. Context free languages (Culik-Salomaa '78)
  2. ET0L languages over 2-letter alphabet (Culik-Richier '79)

The homomorphism equivalence problem

Fix a class of languages $\mathcal{C}$. For a language $\Sigma^* \supseteq L \in \mathcal{C}$ and homomorphisms h, k of Σ^*, the *homomorphism equivalence problem for $\mathcal{C}$*, or **HEP**$(\mathcal{C})$, asks whether $h|_L = k|_L$.

- `INPUT:` $L \in \mathcal{C}$ (e.g. given by a grammar), $h, k : \Sigma^* \to \Sigma^*$
- `OUTPUT:` "YES" if $h|_L = k|_L$; "NO" otherwise.

- Studied by Culik, Salomaa et al. in the 70s
- Decidability shown for:
  1. Context free languages (Culik-Salomaa '78)
  2. ET0L languages over 2-letter alphabet (Culik-Richier '79)
  3. Open for D0L languages until 1985 (see below)

D0L languages:

$L = \{w, f(w), f^2(w), \ldots\}$ for some morphism $f : \Sigma^* \to \Sigma^*$ and start word $w \in \Sigma^*$.

Test sets and the Ehrenfeucht conjecture

One way to solve the **HEP** is by computing a *test set*:

Definition

A *test set* for a language $L \subseteq \Sigma^*$ is a finite subset $T \subseteq L$ s.t. for all homomorphisms $h, k : \Sigma^* \to \Sigma^*$ we have

$$h|_L = k|_L \Leftrightarrow h|_T = k|_T.$$

Test sets and the Ehrenfeucht conjecture

One way to solve the **HEP** is by computing a *test set*:

Definition

A *test set* for a language $L \subseteq \Sigma^*$ is a finite subset $T \subseteq L$ s.t. for all homomorphisms $h, k : \Sigma^* \to \Sigma^*$ we have

$$h|_L = k|_L \Leftrightarrow h|_T = k|_T.$$

Theorem (Ehrenfeucht conjecture; Albert-Lawrence '85, Guba '86)

Every language has a test set...

Test sets and the Ehrenfeucht conjecture

One way to solve the **HEP** is by computing a *test set*:

Definition

A *test set* for a language $L \subseteq \Sigma^*$ is a finite subset $T \subseteq L$ s.t. for all homomorphisms $h, k : \Sigma^* \to \Sigma^*$ we have

$$h|_L = k|_L \Leftrightarrow h|_T = k|_T.$$

Theorem (Ehrenfeucht conjecture; Albert-Lawrence '85, Guba '86)

Every language has a test set...

- ...but they are known to be *computable* in only a few cases, e.g. CF languages (Culik-Salomaa '80)

Test sets and the Ehrenfeucht conjecture

One way to solve the **HEP** is by computing a *test set*:

Definition

A *test set* for a language $L \subseteq \Sigma^*$ is a finite subset $T \subseteq L$ s.t. for all homomorphisms $h, k : \Sigma^* \to \Sigma^*$ we have

$$h|_L = k|_L \Leftrightarrow h|_T = k|_T.$$

Theorem (Ehrenfeucht conjecture; Albert-Lawrence '85, Guba '86)

Every language has a test set...

- ...but they are known to be *computable* in only a few cases, e.g. CF languages (Culik-Salomaa '80)
- test sets computable for $\mathcal{C} \Longrightarrow$ **HEP**$(\mathcal{C})$ is decidable

Test sets and the Ehrenfeucht conjecture

One way to solve the **HEP** is by computing a *test set*:

Definition

A *test set* for a language $L \subseteq \Sigma^*$ is a finite subset $T \subseteq L$ s.t. for all homomorphisms $h, k : \Sigma^* \to \Sigma^*$ we have

$$h|_L = k|_L \Leftrightarrow h|_T = k|_T.$$

Theorem (Ehrenfeucht conjecture; Albert-Lawrence '85, Guba '86)

Every language has a test set...

- ...but they are known to be *computable* in only a few cases, e.g. CF languages (Culik-Salomaa '80)
- test sets computable for $\mathcal{C} \Longrightarrow \mathbf{HEP}(\mathcal{C})$ is decidable
- Open question: does the converse hold?

Ehrenfeucht conjecture proof sketch

Sketch of proof

- Encode Σ^* in two ways: (1) in $\mathrm{SL}_2(\mathbb{Z})$:
  - $a = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right)$, $b = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right)$, $\Sigma = \{A_1, \ldots, A_m\}$
  - $\theta : A_i \mapsto b^{-i} a b^i$

Ehrenfeucht conjecture proof sketch

Sketch of proof

- Encode Σ^* in two ways: (1) in $\mathrm{SL}_2(\mathbb{Z})$:
  - $a = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right)$, $b = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right)$, $\Sigma = \{A_1, \ldots, A_m\}$
  - $\theta : A_i \mapsto b^{-i} a b^i$
- (2) in $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ where $\mathcal{X} = \{X_{ij}, Y_{ij}\}_{1 \leq i \leq m, 1 \leq j \leq 4}$:
  - $\rho_X : A_i \mapsto \left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right)$, $\rho_Y : A_i \mapsto \left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right)$
  - (h, k) encoded as unique map $\sigma_{h,k} : \mathcal{X} \to \mathbb{N}$ s.t.

$$\sigma_{h,k}\left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right) = \theta \circ h(A_i) \text{ and } \sigma_{h,k}\left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right) = \theta \circ k(A_i)$$

Ehrenfeucht conjecture proof sketch

Sketch of proof

- Encode Σ^* in two ways: (1) in $\mathrm{SL}_2(\mathbb{Z})$:
  - $a = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right)$, $b = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right)$, $\Sigma = \{A_1, \ldots, A_m\}$
  - $\theta : A_i \mapsto b^{-i} a b^i$
- (2) in $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ where $\mathcal{X} = \{X_{ij}, Y_{ij}\}_{1 \leq i \leq m, 1 \leq j \leq 4}$:
  - $\rho_X : A_i \mapsto \left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right)$, $\rho_Y : A_i \mapsto \left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right)$
  - (h, k) encoded as unique map $\sigma_{h,k} : \mathcal{X} \to \mathbb{N}$ s.t.

$$\sigma_{h,k}\left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right) = \theta \circ h(A_i) \text{ and } \sigma_{h,k}\left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right) = \theta \circ k(A_i)$$

- poly. equations: $E = \{\rho_X(w) - \rho_Y(w) = \left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right) \mid w \in L\}$

Ehrenfeucht conjecture proof sketch

Sketch of proof

- Encode Σ^* in two ways: (1) in $\mathrm{SL}_2(\mathbb{Z})$:
  - $a = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right)$, $b = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right)$, $\Sigma = \{A_1, \ldots, A_m\}$
  - $\theta : A_i \mapsto b^{-i} a b^i$
- (2) in $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ where $\mathcal{X} = \{X_{ij}, Y_{ij}\}_{1 \le i \le m, 1 \le j \le 4}$:
  - $\rho_X : A_i \mapsto \left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right)$, $\rho_Y : A_i \mapsto \left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right)$
  - (h, k) encoded as unique map $\sigma_{h,k} : \mathcal{X} \to \mathbb{N}$ s.t.

$$\sigma_{h,k}\left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right) = \theta \circ h(A_i) \text{ and } \sigma_{h,k}\left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right) = \theta \circ k(A_i)$$

- poly. equations: $E = \{\rho_X(w) - \rho_Y(w) = \left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right) \mid w \in L\}$
- $h|_L = k|_L \Leftrightarrow \sigma_{h,k}$ is a solution of the system E

Ehrenfeucht conjecture proof sketch

Sketch of proof

- Encode Σ^* in two ways: (1) in $\mathrm{SL}_2(\mathbb{Z})$:
  - $a = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right)$, $b = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right)$, $\Sigma = \{A_1, \ldots, A_m\}$
  - $\theta : A_i \mapsto b^{-i} a b^i$
- (2) in $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ where $\mathcal{X} = \{X_{ij}, Y_{ij}\}_{1 \le i \le m, 1 \le j \le 4}$:
  - $\rho_X : A_i \mapsto \left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right)$, $\rho_Y : A_i \mapsto \left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right)$
  - (h, k) encoded as unique map $\sigma_{h,k} : \mathcal{X} \to \mathbb{N}$ s.t.

$$\sigma_{h,k}\left(\begin{smallmatrix} X_{i1} & X_{i2} \\ X_{i3} & X_{i4} \end{smallmatrix}\right) = \theta \circ h(A_i) \text{ and } \sigma_{h,k}\left(\begin{smallmatrix} Y_{i1} & Y_{i2} \\ Y_{i3} & Y_{i4} \end{smallmatrix}\right) = \theta \circ k(A_i)$$

- poly. equations: $E = \{\rho_X(w) - \rho_Y(w) = \left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right) \mid w \in L\}$
- $h|_L = k|_L \Leftrightarrow \sigma_{h,k}$ is a solution of the system E
- By Hilbert's basis theorem: E has an equiv. finite subsystem.

□

HEP for D0L languages: a novel approach

Let $L = \{w, f(w), f^2(w), \ldots\}$ be a D0L language. We can model f in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation using the substitution

$$\phi : \begin{cases} X_{ij} \mapsto p_{ij} \\ Y_{ij} \mapsto q_{ij} \end{cases}$$

where $p_{ij}, q_{ij} \in \mathbb{Z}[\mathcal{X}]$ are the unique polynomials s.t.

$$\begin{pmatrix} p_{i1} & p_{i2} \\ p_{i3} & p_{i4} \end{pmatrix} = \rho_X(f(A_i)) \text{ and } \begin{pmatrix} q_{i1} & q_{i2} \\ q_{i3} & q_{i4} \end{pmatrix} = \rho_Y(f(A_i)).$$

HEP for D0L languages: a novel approach

Let $L = \{w, f(w), f^2(w), \ldots\}$ be a D0L language. We can model f in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation using the substitution

$$\phi : \begin{cases} X_{ij} \mapsto p_{ij} \\ Y_{ij} \mapsto q_{ij} \end{cases}$$

where $p_{ij}, q_{ij} \in \mathbb{Z}[\mathcal{X}]$ are the unique polynomials s.t.

$$\begin{pmatrix} p_{i1} & p_{i2} \\ p_{i3} & p_{i4} \end{pmatrix} = \rho_X(f(A_i)) \text{ and } \begin{pmatrix} q_{i1} & q_{i2} \\ q_{i3} & q_{i4} \end{pmatrix} = \rho_Y(f(A_i)).$$

Effective construction of a test set for L:

- Form increasing chain $I_1 \subsetneq I_2 \subsetneq \cdots$ of ideals generated by components of $\phi^n(\rho_X(w) - \rho_Y(w))$ for $n = 1, 2, \ldots$
- check strict containment for each n, terminate when stabilized.

HEP for D0L languages: a novel approach

Let $L = \{w, f(w), f^2(w), \ldots\}$ be a D0L language. We can model f in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation using the substitution

$$\phi : \begin{cases} X_{ij} \mapsto p_{ij} \\ Y_{ij} \mapsto q_{ij} \end{cases}$$

where $p_{ij}, q_{ij} \in \mathbb{Z}[\mathcal{X}]$ are the unique polynomials s.t.

$$\begin{pmatrix} p_{i1} & p_{i2} \\ p_{i3} & p_{i4} \end{pmatrix} = \rho_X(f(A_i)) \text{ and } \begin{pmatrix} q_{i1} & q_{i2} \\ q_{i3} & q_{i4} \end{pmatrix} = \rho_Y(f(A_i)).$$

Effective construction of a test set for L:

- Form increasing chain $I_1 \subsetneq I_2 \subsetneq \cdots$ of ideals generated by components of $\phi^n(\rho_X(w) - \rho_Y(w))$ for $n = 1, 2, \ldots$
- check strict containment for each n, terminate when stabilized.

- Question: can we do something similar for indexed languages?

Indexed languages

Fix finite (disjoint) alphabets N of nonterminals and T of terminals, and a start symbol $S \in N$.

Definition

An *indexed grammar* is a tuple $G = (N, T, I, P, S)$ where

- I is a (finite) set of stack letters
- P is a (finite) set of rules of the following types:
  - $A \to Bf$ for $A, B \in N$; $f \in I$
  - $A \to BC$ for $A, B, C \in N$
  - $Af \to B$ for $A, B \in N$; $f \in I$
  - $A \to w$ for $w \in T^*$
- *language of G*: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$

Indexed languages

Fix finite (disjoint) alphabets N of nonterminals and T of terminals, and a start symbol $S \in N$.

Definition

An *indexed grammar* is a tuple $G = (N, T, I, P, S)$ where

- I is a (finite) set of stack letters
- P is a (finite) set of rules of the following types:
  - $A \to Bf$ for $A, B \in N$; $f \in I$
  - $A \to BC$ for $A, B, C \in N$
  - $Af \to B$ for $A, B \in N$; $f \in I$
  - $A \to w$ for $w \in T^*$
- *language of* G: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$

Indexed languages

Fix finite (disjoint) alphabets N of nonterminals and T of terminals, and a start symbol $S \in N$.

Definition

An *indexed grammar* is a tuple $G = (N, T, I, P, S)$ where

- I is a (finite) set of stack letters
- P is a (finite) set of rules of the following types:
  - $A \to Bf$ for $A, B \in N$; $f \in I$
  - $A \to BC$ for $A, B, C \in N$
  - $Af \to B$ for $A, B \in N$; $f \in I$
  - $A \to w$ for $w \in T^*$
- *language of G*: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$

Conjecture (Culik-Salomaa '78)

HEP *is decidable for indexed languages.*

ET0L and EDT0L languages

Definition

- An *ET0L* grammar is a tuple $G = (N, T, \Omega, S)$ where Ω is a set of *tables* (i.e. "nondeterministic morphisms")
- an *EDT0L* grammar is an ET0L grammar where Ω contains only morphisms (D for deterministic)
- the *language of G*: $L(G) = \{w \in T^* \mid w \in \Omega^*(S)\}$
- Language of *sentential forms*: $\mathrm{SF}(G) = \Omega^*(S)$.

ET0L and EDT0L languages

Definition

- An *ET0L* grammar is a tuple $G = (N, T, \Omega, S)$ where Ω is a set of *tables* (i.e. "nondeterministic morphisms")
- an *EDT0L* grammar is an ET0L grammar where Ω contains only morphisms (D for deterministic)
- the *language of G*: $L(G) = \{w \in T^* \mid w \in \Omega^*(S)\}$
- Language of *sentential forms*: $\mathrm{SF}(G) = \Omega^*(S)$.

Inclusion scheme

$$\begin{array}{ccccccc} \mathrm{D0L} & \subset & \mathrm{EDT0L} & \subset & \mathrm{ET0L} & \subset & \text{indexed} \\ & & \cup & & \cup & & \\ & & \mathrm{REG} & \subset & \mathrm{CF} & & \end{array}$$

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

$$\text{Let } \mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$$

be the map which swaps X_{ij} and Y_{ij}.

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism
- Model push$\to$pop derivation trees as follows:

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism
- Model push→pop derivation trees as follows:
  - $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to A_j f) \in P} \psi_f(L_{A_j})$ where ψ_f is the "pop f" homomorphism.

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism
- Model push→pop derivation trees as follows:
  - $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to A_j f) \in P} \psi_f(L_{A_j})$ where ψ_f is the "pop f" homomorphism.
- Model context free production rules as $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to BC) \in P} L_B \cdot L_C$

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism
- Model push→pop derivation trees as follows:
  - $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to A_j f) \in P} \psi_f(L_{A_j})$ where ψ_f is the "pop f" homomorphism.
- Model context free production rules as $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to BC) \in P} L_B \cdot L_C$
- Iteratively update polynomial ideals generated by components of $L_S - \mathcal{XY}(L_S)$

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism
- Model push→pop derivation trees as follows:
  - $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to A_j f) \in P} \psi_f(L_{A_j})$ where ψ_f is the "pop f" homomorphism.
- Model context free production rules as $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to BC) \in P} L_B \cdot L_C$
- Iteratively update polynomial ideals generated by components of $L_S - \mathcal{XY}(L_S)$
- Terminate when ideals stabilize?

HEP for indexed languages (attempt)

Question: can we model indexed grammar production rules in the $M_{2x2}(\mathbb{Z}[\mathcal{X}])$ representation as we did for D0L?

Sketch of a (failed) attempt: Let $\mathcal{XY} : M_{2x2}(\mathbb{Z}[\mathcal{X}]) \to M_{2x2}(\mathbb{Z}[\mathcal{X}])$ be the map which swaps X_{ij} and Y_{ij}.

- Initialize $L_{A_i} = \{\rho_X(A_i)\}$ (destined to be the set of stack-free sentential forms derivable from A_i).
- Assume productiveness and pop-determinism
- Model push→pop derivation trees as follows:
  - $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to A_j f) \in P} \psi_f(L_{A_j})$ where ψ_f is the "pop f" homomorphism.
- Model context free production rules as $L_{A_i} \leftarrow L_{A_i} \cup \bigcup_{(A_i \to BC) \in P} L_B \cdot L_C$
- Iteratively update polynomial ideals generated by components of $L_S - \mathcal{XY}(L_S)$
- Terminate when ideals stabilize? No:(

The good news

However, the D0L procedure extends to EDT0L in a straightforward manner, and ET0L appears promising!

The good news

However, the D0L procedure extends to EDT0L in a straightforward manner, and ET0L appears promising!

Thank you!