Equations over sets of natural numbers.

Artur Jeż

University of Wroclaw

December 13, 2007

Part I

Conjunctive Grammars

Conjunctive grammars

Context-free grammars: Rules of the form

$$A \to \alpha$$

"If w is generated by α, then w is generated by A".

Conjunctive grammars

Context-free grammars: Rules of the form

$$A \to \alpha$$

"If w is generated by α, then w is generated by A".

✓ Multiple rules for A: disjunction.

Conjunctive grammars

Context-free grammars: Rules of the form

$$A \to \alpha$$

"If w is generated by α, then w is generated by A".

✓ Multiple rules for A: disjunction.

Conjunctive grammars (Okhotin, 2000) Rules of the form

$$A \to \alpha_1 \& \ldots \& \alpha_m$$

"If w is generated by each α_i, then w is generated by A".

Definition of conjunctive grammars

- Quadruple $G = (\Sigma, N, P, S)$, where $S \in N$ and rules in P are

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad \text{with } A \in N,\ \alpha_i \in (\Sigma \cup N)^*$$

Definition of conjunctive grammars

- Quadruple $G = (\Sigma, N, P, S)$, where $S \in N$ and rules in P are

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad \text{with } A \in N,\ \alpha_i \in (\Sigma \cup N)^*$$

- Semantics by language equations:

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in P} \ \bigcap_{i=1}^{m} \alpha_i$$

Definition of conjunctive grammars

- Quadruple $G = (\Sigma, N, P, S)$, where $S \in N$ and rules in P are

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad \text{with } A \in N,\ \alpha_i \in (\Sigma \cup N)^*$$

- Semantics by language equations:

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in P} \bigcap_{i=1}^{m} \alpha_i$$

  - $L_G(A)$ is the A-component of the least solution.

Definition of conjunctive grammars

- Quadruple $G = (\Sigma, N, P, S)$, where $S \in N$ and rules in P are

$$A \to \alpha_1 \& \dots \& \alpha_m \quad \text{with } A \in N,\ \alpha_i \in (\Sigma \cup N)^*$$

- Semantics by language equations:

$$A = \bigcup_{A \to \alpha_1 \& \dots \& \alpha_m \in P} \ \bigcap_{i=1}^{m} \alpha_i$$

  - $L_G(A)$ is the A-component of the least solution.

- Semantics by term rewriting:

$$\varphi(A) \Longrightarrow \varphi(\alpha_1 \& \dots \& \alpha_m)$$

Definition of conjunctive grammars

- Quadruple $G = (\Sigma, N, P, S)$, where $S \in N$ and rules in P are

$$A \to \alpha_1 \& \ldots \& \alpha_m \quad \text{with } A \in N,\ \alpha_i \in (\Sigma \cup N)^*$$

- Semantics by language equations:

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in P} \ \bigcap_{i=1}^{m} \alpha_i$$

  - $L_G(A)$ is the A-component of the least solution.

- Semantics by term rewriting:

$$\varphi(A) \Longrightarrow \varphi(\alpha_1 \& \ldots \& \alpha_m)$$
$$\varphi(w \& \ldots \& w) \Longrightarrow \varphi(w)$$

Definition of conjunctive grammars

- Quadruple $G = (\Sigma, N, P, S)$, where $S \in N$ and rules in P are

$$A \to \alpha_1 \& \dots \& \alpha_m \quad \text{with } A \in N,\ \alpha_i \in (\Sigma \cup N)^*$$

- Semantics by language equations:

$$A = \bigcup_{A \to \alpha_1 \& \dots \& \alpha_m \in P} \bigcap_{i=1}^{m} \alpha_i$$

  - $L_G(A)$ is the A-component of the least solution.

- Semantics by term rewriting:

$$\varphi(A) \Longrightarrow \varphi(\alpha_1 \& \dots \& \alpha_m)$$
$$\varphi(w \& \dots \& w) \Longrightarrow \varphi(w)$$

  - $L_G(A) = \{w \mid A \Longrightarrow^* w\}$

Part II

Unary alphabet and equations over sets of numbers

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

- a^n $\longleftrightarrow$ number n

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

- a^n $\longleftrightarrow$ number n
- $a^n \cdot a^m$ $\longleftrightarrow$ $n + m$

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

- a^n $\longleftrightarrow$ number n
- $a^n \cdot a^m$ $\longleftrightarrow$ $n + m$
- Language $\longleftrightarrow$ set of numbers

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

- $a^n \longleftrightarrow$ number n
- $a^n \cdot a^m \longleftrightarrow n + m$
- Language $\longleftrightarrow$ set of numbers
- $K \cdot L \longleftrightarrow X \boxplus Y = \{x + y \mid x \in X, y \in Y\}$

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

- a^n $\longleftrightarrow$ number n
- $a^n \cdot a^m$ $\longleftrightarrow$ $n + m$
- Language $\longleftrightarrow$ set of numbers
- $K \cdot L$ $\longleftrightarrow$ $X \boxplus Y = \{x + y \mid x \in X, y \in Y\}$
- Language equations $\longleftrightarrow$ Equations over subsets of $\mathbb{N}$

The case of a unary alphabet

Unary: $\Sigma = \{a\}$.

- $a^n \longleftrightarrow$ number n
- $a^n \cdot a^m \longleftrightarrow n + m$
- Language $\longleftrightarrow$ set of numbers
- $K \cdot L \longleftrightarrow X \boxplus Y = \{x + y \mid x \in X, y \in Y\}$
- Language equations $\longleftrightarrow$ Equations over subsets of $\mathbb{N}$

Resolved equations over sets of natural numbers.

Equations over sets of numbers

$$\begin{cases} X_1 & = & \varphi_1(X_1, \ldots, X_n) \\ & \vdots & \\ X_n & = & \varphi_n(X_1, \ldots, X_n) \end{cases}$$

Equations over sets of numbers

$$\begin{cases} X_1 & = & \varphi_1(X_1, \ldots, X_n) \\ & \vdots & \\ X_n & = & \varphi_n(X_1, \ldots, X_n) \end{cases}$$

- X_i: subset of $\mathbb{N} = \{0, 1, 2, \ldots\}$.

Equations over sets of numbers

$$\begin{cases} X_1 &= \varphi_1(X_1, \dots, X_n) \\ & \vdots \\ X_n &= \varphi_n(X_1, \dots, X_n) \end{cases}$$

- X_i: subset of $\mathbb{N} = \{0, 1, 2, \dots\}$.
- φ_i contains variables, singleton constants, operations on sets.

Equations over sets of numbers

$$\begin{cases} X_1 &= \varphi_1(X_1, \dots, X_n) \\ &\vdots \\ X_n &= \varphi_n(X_1, \dots, X_n) \end{cases}$$

- X_i: subset of $\mathbb{N} = \{0, 1, 2, \dots\}$.
- φ_i contains variables, singleton constants, operations on sets.
- Operations : $\cup, \cap, \boxplus$

Definition

$$X \boxplus Y = \{x + y : x \in X,\ y \in Y\}$$

Equations over sets of numbers

$$\begin{cases} X_1 & = & \varphi_1(X_1, \ldots, X_n) \\ & \vdots & \\ X_n & = & \varphi_n(X_1, \ldots, X_n) \end{cases}$$

- X_i: subset of $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- φ_i contains variables, singleton constants, operations on sets.
- Operations : $\cup, \cap, \boxplus$

Definition

$$X \boxplus Y = \{x + y : x \in X,\ y \in Y\}$$

- Example:

$$X = (X \boxplus X) \cup \{2\}$$

Equations over sets of numbers

$$\begin{cases} X_1 &= \varphi_1(X_1, \ldots, X_n) \\ &\vdots \\ X_n &= \varphi_n(X_1, \ldots, X_n) \end{cases}$$

- X_i: subset of $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- φ_i contains variables, singleton constants, operations on sets.
- Operations : $\cup, \cap, \boxplus$

Definition

$$X \boxplus Y = \{x + y : x \in X,\ y \in Y\}$$

- Example:

$$X = (X \boxplus X) \cup \{2\}$$

- $EQ(\cup, \cap, \boxplus)$—sets expressible as least solutions

Questions

Problem

✓ *Expressive power?*

Questions

Problem

- ✓ *Expressive power?*
- ✓ *Complexity of the membership problem?*

Questions

Problem

- ✓ *Expressive power?*
- ✓ *Complexity of the membership problem?*

Remark

Operations $\{\cup, \boxplus\}$:

Questions

Problem

✓ *Expressive power?*

✓ *Complexity of the membership problem?*

Remark

Operations $\{\cup, \boxplus\}$:

- Context-free grammars over an alphabet $\{a\}$.

Questions

Problem

✓ *Expressive power?*

✓ *Complexity of the membership problem?*

Remark

Operations $\{\cup, \boxplus\}$:

- Context-free grammars over an alphabet $\{a\}$.
- Least solutions are ultimately periodic.

Questions

Problem

✓ *Expressive power?*

✓ *Complexity of the membership problem?*

Remark

Operations $\{\cup, \boxplus\}$:

- Context-free grammars over an alphabet $\{a\}$.
- Least solutions are ultimately periodic.
- General membership problem: NP-complete

Tool: positional notation

- Using base-k notation.

Tool: positional notation

- Using base-k notation.
- $\Sigma_k = \{0, 1, \ldots, k-1\}$.

Tool: positional notation

- Using base-k notation.
- $\Sigma_k = \{0, 1, \ldots, k-1\}$.
- Numbers $\longleftrightarrow$ strings in $\Sigma_k^* \setminus 0\Sigma_k^*$.

Tool: positional notation

- Using base-k notation.
- $\Sigma_k = \{0, 1, \ldots, k-1\}$.
- Numbers $\longleftrightarrow$ strings in $\Sigma_k^* \setminus 0\Sigma_k^*$.
- Sets of numbers $\longleftrightarrow$ formal languages over Σ_k.

Tool: positional notation

- Using base-k notation.
- $\Sigma_k = \{0, 1, \ldots, k-1\}$.
- Numbers $\longleftrightarrow$ strings in $\Sigma_k^* \setminus 0\Sigma_k^*$.
- Sets of numbers $\longleftrightarrow$ formal languages over Σ_k.

Example

$$(10^*)_4 = \{4^n \mid n \geqslant 0\}$$

Example of non-periodic solution ($k = 4$)

Solution

$$\begin{aligned} L_1 &= 10^* , \\ L_2 &= 20^* , \\ L_3 &= 30^* , \\ L_{12} &= 120^* . \end{aligned}$$

Example of non-periodic solution ($k = 4$)

Solution

$$
\begin{aligned}
L_1 &= 10^* , \\
L_2 &= 20^* , \\
L_3 &= 30^* , \\
L_{12} &= 120^* .
\end{aligned}
$$

Equations

$$
\begin{aligned}
B_1 &= (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\} , \\
B_2 &= (B_{12} \boxplus B_2 \cap B_1 \boxplus B_1) \cup \{2\} , \\
B_3 &= (B_{12} \boxplus B_{12} \cap B_1 \boxplus B_2) \cup \{3\} , \\
B_{12} &= (B_3 \boxplus B_3 \cap B_1 \boxplus B_2) .
\end{aligned}
$$

What needs to be proved

- By general knowledge there is a unique ε-free solution.

What needs to be proved

- By general knowledge there is a unique ε-free solution.
- Vector of sets $(\ldots, \mathtt{i0}^*, \ldots)$ is ε-free.

What needs to be proved

- By general knowledge there is a unique ε-free solution.
- Vector of sets $(\ldots, \mathtt{i0}^*, \ldots)$ is ε-free.
- We need to show that it is a solution.

What needs to be proved

- By general knowledge there is a unique ε-free solution.
- Vector of sets $(\dots, \mathtt{i0}^*, \dots)$ is ε-free.
- We need to show that it is a solution.

Example

For example 10^*, the rule is

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

So we want to prove that

$$10^* = 20^* \boxplus 20^* \cap 10^* \boxplus 30^* \cup \{1\}$$

Calculations

Rule:

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

Calculations

Rule:

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

Proof.

$$20^* \boxplus 20^* \quad = \quad 10^+ \cup 20^* 20^*$$

Calculations

Rule:

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

Proof.

$$\begin{aligned} 20^* \boxplus 20^* &= 10^+ \cup 20^*20^* \\ 10^* \boxplus 30^* &= 10^+ \cup 10^*30^* \cup 30^*10^* \end{aligned}$$

Calculations

Rule:

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

Proof.

$$\begin{aligned} 20^* \boxplus 20^* &= 10^+ \cup 20^* 20^* \\ 10^* \boxplus 30^* &= 10^+ \cup 10^* 30^* \cup 30^* 10^* \\ 20^* \boxplus 20^* \cap 10^* \boxplus 30^* &= 10^+ \end{aligned}$$

Calculations

Rule:

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

Proof.

$$\begin{aligned}
20^* \boxplus 20^* &= 10^+ \cup 20^*20^* \\
10^* \boxplus 30^* &= 10^+ \cup 10^*30^* \cup 30^*10^* \\
20^* \boxplus 20^* \cap 10^* \boxplus 30^* &= 10^+ \\
20^* \boxplus 20^* \cap 10^* \boxplus 30^* \cup \{1\} &= 10^*
\end{aligned}$$

Calculations

Rule:

$$B_1 = (B_2 \boxplus B_2 \cap B_1 \boxplus B_3) \cup \{1\}$$

Proof.

$$\begin{aligned}
20^* \boxplus 20^* &= 10^+ \cup 20^* 20^* \\
10^* \boxplus 30^* &= 10^+ \cup 10^* 30^* \cup 30^* 10^* \\
20^* \boxplus 20^* \cap 10^* \boxplus 30^* &= 10^+ \\
20^* \boxplus 20^* \cap 10^* \boxplus 30^* \cup \{1\} &= 10^*
\end{aligned}$$

□

Remark

Similar proof for $ij0^*$ in base-k notation.

Any regular language

Theorem (Jeż, DLT 2007)

For every k and $R \subset \{0, \dots, k-1\}^*$ if R is regular then $R \in EQ(\cap, \cup, \boxplus)$.

Any regular language

Theorem (Jeż, DLT 2007)

For every k and $R \subset \{0, \ldots, k-1\}^*$ if R is regular then $R \in EQ(\cap, \cup, \boxplus)$.

Idea

Let $\langle \{0, \ldots, k-1\}, Q, q_0, F, \delta \rangle$ recognize R.

Any regular language

Theorem (Jeż, DLT 2007)

For every k and $R \subset \{0, \dots, k-1\}^*$ if R is regular then $R \in EQ(\cap, \cup, \boxplus)$.

Idea

Let $\langle \{0, \dots, k-1\}, Q, q_0, F, \delta \rangle$ recognize R.
We introduce variable $B_{i,j,q}$ for set

$$\{ijw : \delta(q_0, w) = q\}$$

Any regular language

Theorem (Jeż, DLT 2007)

For every k and $R \subset \{0, \ldots, k-1\}^*$ if R is regular then $R \in EQ(\cap, \cup, \boxplus)$.

Idea

Let $\langle \{0, \ldots, k-1\}, Q, q_0, F, \delta \rangle$ recognize R.
We introduce variable $B_{i,j,q}$ for set

$$\{ijw : \delta(q_0, w) = q\}$$

Information the indices carry:

- *leading symbol i*
- *second leading symbol j*
- *q—the computation of M on the rest of the word*

Equations for $B_{i,j,q}$

Example

$$B_{i,j,q} = \bigcup_{(x,q'):q\in\delta(q',x)} \bigcap_{n=1}^{4} B_{i-1,j+n} \boxplus B_{k-n,x,q'} \cup \ldots$$

Equations for $B_{i,j,q}$

Example

$$B_{i,j,q} = \bigcup_{(x,q'):q\in\delta(q',x)} \bigcap_{n=1}^{4} B_{i-1,j+n} \boxplus B_{k-n,x,q'} \cup \dots$$

$$\begin{array}{ccccc} & & k-n & x & \overbrace{\ldots}^{\text{state } q'} \\ + & i-1 & j+n & & 00\ldots0 \\ \hline & i & j & & \underbrace{x\ldots\ldots}_{\text{state } q} \end{array}$$

Trellis automata

(one-way real-time cellular automata)

Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Trellis automata

(one-way real-time cellular automata)

Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a
$M = (\Sigma, Q, I, \delta, F)$ where:

Trellis automata

(one-way real-time cellular automata)

Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;

Trellis automata

(one-way real-time cellular automata)

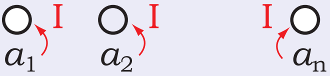
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;



Trellis automata

(one-way real-time cellular automata)

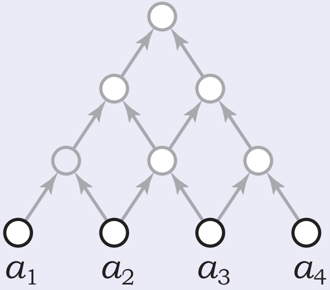
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;



Trellis automata

(one-way real-time cellular automata)

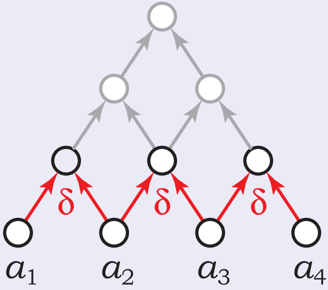
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ *trellis automaton* M *over* Σ_k *with* $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$,
set $L(M)$ *is in* $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a
$M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;
- $\delta : Q \times Q \to Q$, transition function;



Trellis automata

(one-way real-time cellular automata)

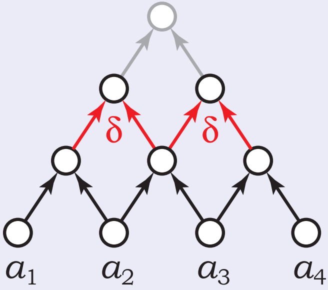
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ *trellis automaton* M *over* Σ_k *with* $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, *set* $L(M)$ *is in* $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;
- $\delta : Q \times Q \to Q$, transition function;



Trellis automata

(one-way real-time cellular automata)

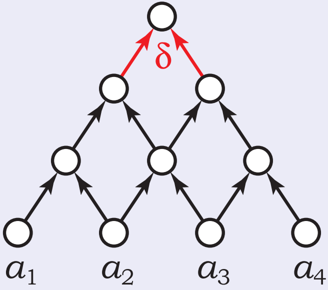
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ *trellis automaton* M *over* Σ_k *with* $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$,
set $L(M)$ *is in* $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;
- $\delta : Q \times Q \to Q$, transition function;



Trellis automata

(one-way real-time cellular automata)

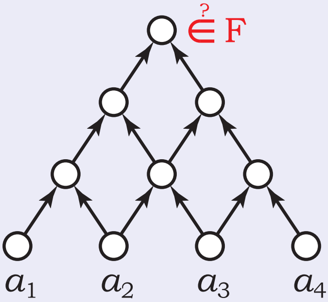
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;
- $\delta : Q \times Q \to Q$, transition function;
- $F \subseteq Q$: accepting states.



Trellis automata

(one-way real-time cellular automata)

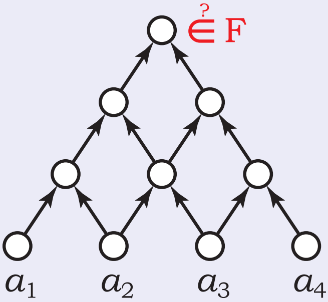
Theorem (Jeż, Okhotin, CSR 2007)

$\forall$ trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, set $L(M)$ is in $EQ(\cap, \cup, \boxplus)$.

Definition (Culik, Gruska, Salomaa, 1981)

A trellis automaton is a $M = (\Sigma, Q, I, \delta, F)$ where:

- Σ: input alphabet;
- Q: finite set of states;
- $I : \Sigma \to Q$ sets initial states;
- $\delta : Q \times Q \to Q$, transition function;
- $F \subseteq Q$: accepting states.



- Closed under $\cup, \cap, \sim$, not closed under concatenation.

Main lemma

Lemma

For every trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, there exists a system of equations in $EQ = (\cup, \cap, \boxplus)$ with least solution

$$\{1w10^* \mid w + 1 \in L(M)\}, \ldots,$$

Main lemma

Lemma

For every trellis automaton M over Σ_k with $L(M) \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$, there exists a system of equations in $EQ = (\cup, \cap, \boxplus)$ with least solution

$$\{1w10^* \mid w + 1 \in L(M)\}, \ldots,$$

- $1w10^*$ represents w.

The construction

- Set of variables $\{X_q \mid q \in Q\}$.

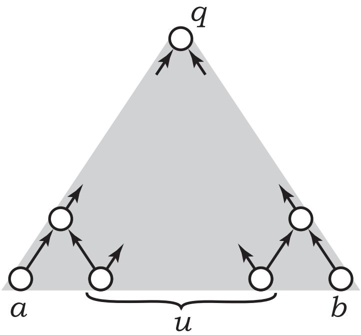
The construction

- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$

The construction

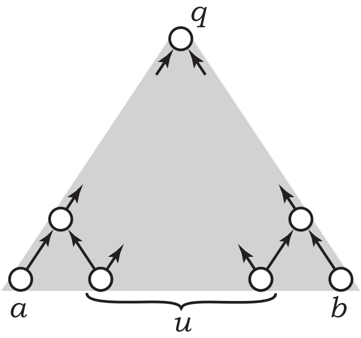
- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$

- $aub \in L_M(q)$



The construction

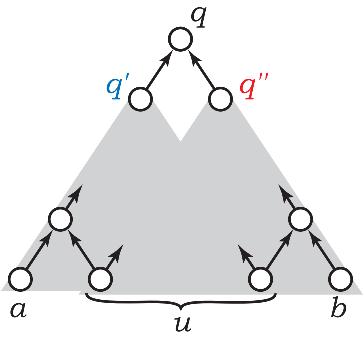
- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$
- $aub \in L_M(q) \Leftrightarrow$



The construction

- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$

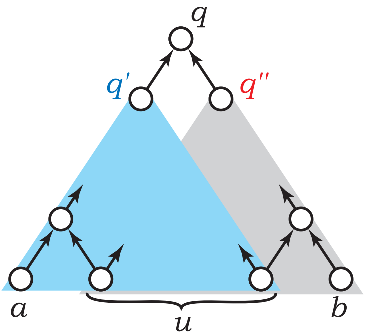
- $aub \in L_M(q) \Leftrightarrow$
  $\exists q', q'' : \delta(q', q'') = q$,



The construction

- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$

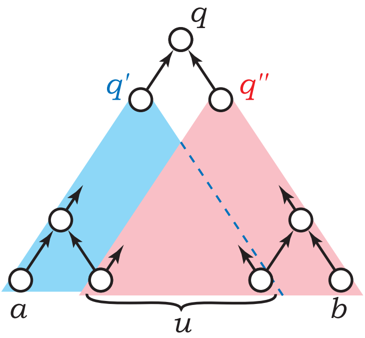
- $aub \in L_M(q) \Leftrightarrow$
  $\exists q', q'' : \delta(q', q'') = q$,
  $au \in L_M(q')$,



The construction

- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$

- $aub \in L_M(q) \Leftrightarrow$
  $\exists q', q'' : \delta(q', q'') = q$,
  $au \in L_M(q')$,
  $ub \in L_M(q'')$.

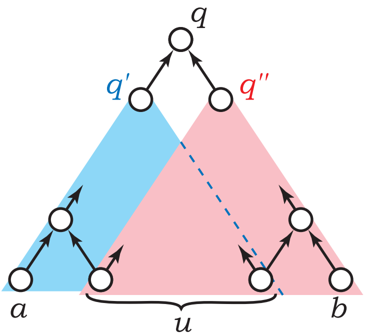


The construction

- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w + 1 \in L_M(q)\}$

- $aub \in L_M(q) \Leftrightarrow$
  $\exists q', q'' : \delta(q', q'') = q$,
  $au \in L_M(q')$,
  $ub \in L_M(q'')$.
- Let $1au10^* \subseteq X_{q'}$, $1ub10^* \subseteq X_{q''}$.

$$X_q = \bigcup_{\substack{q',q'':\delta(q',q'')=q \\ a,b\in\Sigma_k}} \rho_b(X_{q'}) \cap \lambda_a(X_{q''})$$

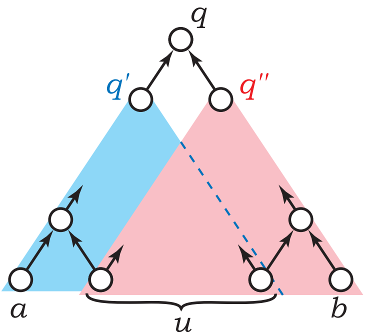


The construction

- Set of variables $\{X_q \mid q \in Q\}$.
- Actually, $X_q = \{1w10^* \mid w+1 \in L_M(q)\}$

- $aub \in L_M(q) \Leftrightarrow$
  $\exists q', q'' : \delta(q', q'') = q$,
  $au \in L_M(q')$,
  $ub \in L_M(q'')$.
- Let $1au10^* \subseteq X_{q'}$, $1ub10^* \subseteq X_{q''}$.

$$X_q = \bigcup_{\substack{q',q'':\delta(q',q'')=q \\ a,b\in\Sigma_k}} \rho_b(X_{q'}) \cap \lambda_a(X_{q''})$$



$$\lambda_a(1w10^k) = 1aw10^k$$

$$\rho_b(1w10^k) = 1wb10^{k-1}$$

Part III

Complexity of equations with $\{\cup, \cap, \boxplus\}$

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.
- Space complexity: using $s(n)$ elementary memory cells.

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.
- Space complexity: using $s(n)$ elementary memory cells.

P polynomial time.

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.
- Space complexity: using $s(n)$ elementary memory cells.

P polynomial time.

NP nondeterministic polynomial time (may guess).

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.
- Space complexity: using $s(n)$ elementary memory cells.

P	polynomial time.
NP	nondeterministic polynomial time (may guess).
PSPACE	polynomial space.

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.
- Space complexity: using $s(n)$ elementary memory cells.

P	polynomial time.
NP	nondeterministic polynomial time (may guess).
PSPACE	polynomial space.
EXPTIME	exponential time.

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

Computational complexity: basic notions

- Fix $X \subseteq \mathbb{N}_0$.
- Determine algorithmically whether $x \in X$.
- $n = \log x$: length of notation of x
- Time complexity: in $t(n)$ elementary steps.
- Space complexity: using $s(n)$ elementary memory cells.

P	polynomial time.
NP	nondeterministic polynomial time (may guess).
PSPACE	polynomial space.
EXPTIME	exponential time.

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

- C-complete set X: every problem in C can be reduced to X.

Complexity of solutions

- Trellis automata recognize P-complete languages.

Complexity of solutions

- Trellis automata recognize P-complete languages.
- P-complete sets of numbers.

Complexity of solutions

- Trellis automata recognize P-complete languages.
- P-complete sets of numbers.
- NP-complete sets: relatively easy.

Complexity of solutions

- Trellis automata recognize P-complete languages.
- P-complete sets of numbers.
- NP-complete sets: relatively easy.
- PSPACE-complete sets: requires some efforts.

Complexity of solutions

- Trellis automata recognize P-complete languages.
- P-complete sets of numbers.
- NP-complete sets: relatively easy.
- PSPACE-complete sets: requires some efforts.
- Upper bound:

Theorem (Okhotin, 2001)

Every conjunctive language can be recognized in time $O(n^3)$.

Complexity of solutions

- Trellis automata recognize P-complete languages.
- P-complete sets of numbers.
- NP-complete sets: relatively easy.
- PSPACE-complete sets: requires some efforts.
- Upper bound:

Theorem (Okhotin, 2001)

Every conjunctive language can be recognized in time $O(n^3)$.

Corollary

Every set of numbers in $EQ(\cup, \cap, \boxplus)$ is in EXPTIME.

Complexity of solutions

- Trellis automata recognize P-complete languages.
- P-complete sets of numbers.
- NP-complete sets: relatively easy.
- PSPACE-complete sets: requires some efforts.
- Upper bound:

Theorem (Okhotin, 2001)

Every conjunctive language can be recognized in time $O(n^3)$.

Corollary

Every set of numbers in $EQ(\cup, \cap, \boxplus)$ is in EXPTIME.

Theorem (Jeż, Okhotin, STACS 2008)

$EQ(\cup, \cap, \boxplus)$ *contains an EXPTIME-complete set.*

Alternating Turing machines

- Tape alphabet Γ, set of states $Q = Q_E \cup Q_A \cup \{q_{acc}\}$.

Alternating Turing machines

- Tape alphabet Γ, set of states $Q = Q_E \cup Q_A \cup \{q_{acc}\}$.
- Transition function $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\leftarrow, \downarrow, \rightarrow\}}$.

Alternating Turing machines

- Tape alphabet Γ, set of states $Q = Q_E \cup Q_A \cup \{q_{acc}\}$.
- Transition function $\delta : Q \times \Gamma \to 2^{Q\times\Gamma\times\{\leftarrow,\downarrow,\rightarrow\}}$.
- If $q = q_{acc}$, accepts from here.

Alternating Turing machines

- Tape alphabet Γ, set of states $Q = Q_E \cup Q_A \cup \{q_{acc}\}$.
- Transition function $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\leftarrow,\downarrow,\rightarrow\}}$.
- If $q = q_{acc}$, accepts from here.
- If $q \in Q_E$, accepts from here if accepts from some next conf.

Alternating Turing machines

- Tape alphabet Γ, set of states $Q = Q_E \cup Q_A \cup \{q_{acc}\}$.
- Transition function $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\leftarrow, \downarrow, \rightarrow\}}$.
- If $q = q_{acc}$, accepts from here.
- If $q \in Q_E$, accepts from here if accepts from some next conf.
- If $q \in Q_A$, accepts from here if accepts from every next conf.

Alternating Turing machines

- Tape alphabet Γ, set of states $Q = Q_E \cup Q_A \cup \{q_{acc}\}$.
- Transition function $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\leftarrow, \downarrow, \rightarrow\}}$.
- If $q = q_{acc}$, accepts from here.
- If $q \in Q_E$, accepts from here if accepts from some next conf.
- If $q \in Q_A$, accepts from here if accepts from every next conf.

Theorem (A. Chandra, D. Kozen, L. Stockmeyer 1981)

$$APSPACE = EXPTIME$$
$$APTIME = PSPACE$$

Idea of encoding

Problem

How to encode a configuration?

Idea of encoding

Problem

How to encode a configuration?

Idea

- *Arithmetization of a configuration*

Idea of encoding

Problem

How to encode a configuration?

Idea

- *Arithmetization of a configuration*
- *Define final accepting configurations*

Idea of encoding

Problem

How to encode a configuration?

Idea

- *Arithmetization of a configuration*
- *Define final accepting configurations*
- *Calculate previous accepting configurations*

Idea of encoding

Problem

How to encode a configuration?

Idea

- *Arithmetization of a configuration*
- *Define final accepting configurations*
- *Calculate previous accepting configurations*
- *Alternation is not a problem*

Idea of encoding

Problem

How to encode a configuration?

Idea

- *Arithmetization of a configuration*
- *Define final accepting configurations*
- *Calculate previous accepting configurations*
- *Alternation is not a problem*
- *Problem: numbers increase with every step, encodings not*

Idea of encoding

Problem

How to encode a configuration?

Idea

- *Arithmetization of a configuration*
- *Define final accepting configurations*
- *Calculate previous accepting configurations*
- *Alternation is not a problem*
- *Problem: numbers increase with every step, encodings not*
- *Solution: restricting the model and adding a counter*

Restrictions of the model

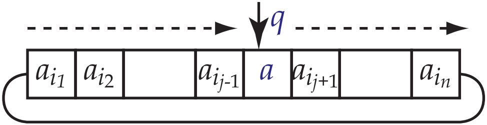
- Circular tape.

Restrictions of the model

- Circular tape.
- Moving to the right at every step.

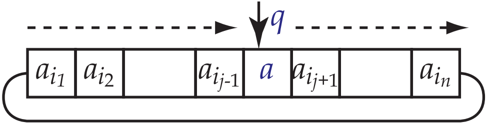
Restrictions of the model

- Circular tape.
- Moving to the right at every step.
- Next configuration:



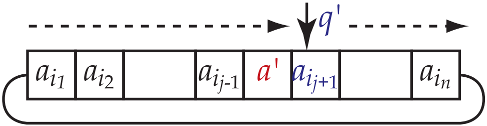
Restrictions of the model

- Circular tape.
- Moving to the right at every step.
- Next configuration: $(q', a') \in \delta(q, a)$



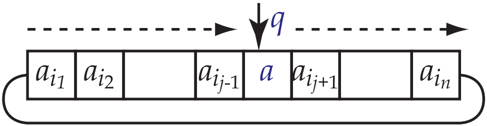
Restrictions of the model

- Circular tape.
- Moving to the right at every step.
- Next configuration: $(q', a') \in \delta(q, a)$



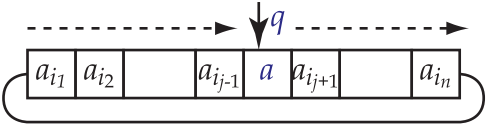
Restrictions of the model

- Circular tape.
- Moving to the right at every step.
- Next configuration:



Restrictions of the model

- Circular tape.
- Moving to the right at every step.
- Next configuration:



Remark

Still $APSPACE = EXPTIME$.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \ldots, a_{|\Gamma|-1}\}$.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \ldots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \ldots, k-1\}$.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \dots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \dots, k-1\}$.
- $\langle\cdot\rangle : Q \cup \Gamma \to \Sigma$.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \dots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \dots, k-1\}$.
- $\langle\cdot\rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \ldots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \ldots, k-1\}$.
- $\langle \cdot \rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \dots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \dots, k-1\}$.
- $\langle \cdot \rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,
- Instantaneous description:

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \ldots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \ldots, k-1\}$.
- $\langle\cdot\rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,
- Instantaneous description:
  - Tape containing $a_{i_1} \ldots a_{i_n}$

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \dots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \dots, k-1\}$.
- $\langle\cdot\rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,
- Instantaneous description:
  - Tape containing $a_{i_1} \dots a_{i_n}$
  - In state q over a_{i_j}.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \ldots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \ldots, k-1\}$.
- $\langle\cdot\rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,
- Instantaneous description:
  - Tape containing $a_{i_1} \ldots a_{i_n}$
  - In state q over a_{i_j}.
  - At most r rotations over the tape, with $r = \sum_{i=0}^{\ell} 2^i c_i$, $c_i \in \{0, 1\}$.

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \ldots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \ldots, k-1\}$.
- $\langle \cdot \rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,
- Instantaneous description:
  - Tape containing $a_{i_1} \ldots a_{i_n}$
  - In state q over a_{i_j}.
  - At most r rotations over the tape, with $r = \sum_{i=0}^{\ell} 2^i c_i$, $c_i \in \{0, 1\}$.
- As a number in base-k notation:

$$\underbrace{1 c_{\ell-1} \ldots c_1 c_0}_{\textit{counter}} 55 \underbrace{0\langle a_{i_1} \rangle \ldots 0\langle a_{i_{j-1}} \rangle \langle q \rangle \langle a_{i_j} \rangle 0 \langle a_{i_{j+1}} \rangle \ldots 0 \langle a_{i_n} \rangle 0}_{\textit{tape}} \in \Sigma^*$$

Arithmetization of alternating Turing machines

- Tape alphabet $\Gamma = \{a_0, \dots, a_{|\Gamma|-1}\}$.
- Let $k = 8 + |Q| + \max(|Q| + 7, |\Gamma|)$, let $\Sigma = \{0, \dots, k-1\}$.
- $\langle\cdot\rangle : Q \cup \Gamma \to \Sigma$.
  - $\langle q_i \rangle = 7 + i$ for $q_i \in Q$.
  - $\langle a_j \rangle = 7 + |Q| + j$ for $a_i \in \Gamma$,
- Instantaneous description:
  - Tape containing $a_{i_1} \dots a_{i_n}$
  - In state q over a_{i_j}.
  - At most r rotations over the tape, with $r = \sum_{i=0}^{\ell} 2^i c_i$, $c_i \in \{0, 1\}$.
- As a number in base-k notation:

$$\underbrace{1 c_{\ell-1} \dots c_1 c_0}_{\textit{counter}} 55 \underbrace{0\langle a_{i_1} \rangle \dots 0\langle a_{i_{j-1}} \rangle \langle q \rangle \langle a_{i_j} \rangle 0 \langle a_{i_{j+1}} \rangle \dots 0 \langle a_{i_n} \rangle 0}_{\textit{tape}} \in \Sigma^*$$

- Decreases at every step of computation.

Constructing equations: lower level

- $\text{Move}_{q',a',q,a}(X)$: transition of the ATM.

Constructing equations: lower level

- $\text{Move}_{q',a',q,a}(X)$: transition of the ATM.
- $\text{Move}_{q',a',q,a}(X)$ contains all IDs

$$1c_{\ell-1}\dots c_1c_0550\langle a_{i_1}\rangle\dots 0\langle a_{i_{j-1}}\rangle\langle q\rangle\langle a\rangle 0\langle a_{i_{j+1}}\rangle\dots 0\langle a_{i_n}\rangle 0,$$

for which

Constructing equations: lower level

- $\mathrm{Move}_{q',a',q,a}(X)$: transition of the ATM.
- $\mathrm{Move}_{q',a',q,a}(X)$ contains all IDs

$$1c_{\ell-1}\dots c_1c_0 550\langle a_{i_1}\rangle\dots 0\langle a_{i_{j-1}}\rangle\langle q\rangle\langle a\rangle 0\langle a_{i_{j+1}}\rangle\dots 0\langle a_{i_n}\rangle 0,$$

for which

$$1c_{\ell-1}\dots c_1c_0 550\langle a_{i_1}\rangle\dots 0\langle a_{i_{j-1}}\rangle 0\langle a'\rangle\langle q'\rangle\langle a_{i_{j+1}}\rangle\dots 0\langle a_{i_n}\rangle 0 \in X$$

Constructing equations: lower level

- $\text{Move}_{q',a',q,a}(X)$: transition of the ATM.
- $\text{Move}_{q',a',q,a}(X)$ contains all IDs

$$1c_{\ell-1}\dots c_1c_0550\langle a_{i_1}\rangle\dots 0\langle a_{i_{j-1}}\rangle\langle q\rangle\langle a\rangle 0\langle a_{i_{j+1}}\rangle\dots 0\langle a_{i_n}\rangle 0,$$

  for which

$$1c_{\ell-1}\dots c_1c_0550\langle a_{i_1}\rangle\dots 0\langle a_{i_{j-1}}\rangle 0\langle a'\rangle\langle q'\rangle\langle a_{i_{j+1}}\rangle\dots 0\langle a_{i_n}\rangle 0\in X$$

- Equation:

$$\begin{aligned}\text{Move}_{q,a,q',a'}(X) = \big(X \cap \text{Counter}\,55\,\text{Tape}_{q'a'}\big)&\\ \boxplus\big(\langle q\rangle\langle a\rangle 0 \boxminus \langle a'\rangle\langle q'\rangle\big)(00)^*&\\ \cap\,\text{Counter}\,55\,\text{Tape}_{aq}&\end{aligned}$$

Constructing equations: upper level

$$X = \mathrm{Final} \cup \mathrm{Step}(X) \cup \left(Y \cap \mathrm{Counter}\,55\,\mathrm{Tape} \right)$$
$$Y = \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)$$

- Final: the set of accepting configurations.

Constructing equations: upper level

$$
\begin{aligned}
X &= \mathrm{Final} \cup \mathrm{Step}(X) \cup \left(Y \cap \mathrm{Counter\,55\,Tape} \right) \\
Y &= \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)
\end{aligned}
$$

- Final: the set of accepting configurations.
- Counter 55 Tape: the set of valid IDs.

Constructing equations: upper level

$$X = \mathrm{Final} \cup \mathrm{Step}(X) \cup \big(Y \cap \mathrm{Counter\,55\,Tape}\big)$$
$$Y = \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)$$

- Final: the set of accepting configurations.
- Counter 55 Tape: the set of valid IDs.
- $\mathrm{Step}(X) = \{n \mid \exists m \in X : \ m \vdash n\}$: to the next square.

Constructing equations: upper level

$$X = \mathrm{Final} \cup \mathrm{Step}(X) \cup \big(Y \cap \mathrm{Counter\,55\,Tape} \big)$$
$$Y = \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)$$

- Final: the set of accepting configurations.
- Counter 55 Tape: the set of valid IDs.
- $\mathrm{Step}(X) = \{n \mid \exists m \in X : \ m \vdash n\}$: to the next square.
- $\mathrm{Jump}(X) = \{n \mid \exists m \in X : \ m \vdash' n\}$: to the first symbol.

Constructing equations: upper level

$$X = \mathrm{Final} \cup \mathrm{Step}(X) \cup \big(Y \cap \mathrm{Counter\,55\,Tape}\big)$$
$$Y = \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)$$

- Final: the set of accepting configurations.
- Counter 55 Tape: the set of valid IDs.
- $\mathrm{Step}(X) = \{n \mid \exists m \in X : \ m \vdash n\}$: to the next square.
- $\mathrm{Jump}(X) = \{n \mid \exists m \in X : \ m \vdash' n\}$: to the first symbol.
- $\mathrm{Carry}(X)$: processing the carry in the counter.

Constructing equations: upper level

$$
\begin{aligned}
X &= \mathrm{Final} \cup \mathrm{Step}(X) \cup \big(Y \cap \mathrm{Counter\,55\,Tape} \big) \\
Y &= \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)
\end{aligned}
$$

- Final: the set of accepting configurations.
- Counter 55 Tape: the set of valid IDs.
- $\mathrm{Step}(X) = \{n \mid \exists m \in X : \ m \vdash n\}$: to the next square.
- $\mathrm{Jump}(X) = \{n \mid \exists m \in X : \ m \vdash' n\}$: to the first symbol.
- $\mathrm{Carry}(X)$: processing the carry in the counter.

$$
\mathrm{Step}(X) = \Big(\bigcup_{q \in Q_E, a \in \Gamma} \ \bigcup_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X) \Big) \cup
$$

$$
\cup
$$

Constructing equations: upper level

$$X = \mathrm{Final} \cup \mathrm{Step}(X) \cup \big(Y \cap \mathrm{Counter\,55\,Tape}\big)$$
$$Y = \mathrm{Jump}(X) \cup \mathrm{Carry}(Y)$$

- Final: the set of accepting configurations.
- Counter 55 Tape: the set of valid IDs.
- $\mathrm{Step}(X) = \{n \mid \exists m \in X : \ m \vdash n\}$: to the next square.
- $\mathrm{Jump}(X) = \{n \mid \exists m \in X : \ m \vdash' n\}$: to the first symbol.
- $\mathrm{Carry}(X)$: processing the carry in the counter.

$$\mathrm{Step}(X) = \Big(\bigcup_{q \in Q_E, a \in \Gamma} \ \bigcup_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X)\Big) \cup$$
$$\cup \Big(\bigcup_{q \in Q_A, a \in \Gamma} \ \bigcap_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X)\Big)$$

Conclusion

- A basic mathematical object.

Conclusion

- A basic mathematical object.
- Using methods of theoretical computer science.

Conclusion

- A basic mathematical object.
- Using methods of theoretical computer science.
- High expressive power and hard recognition

Conclusion

- A basic mathematical object.
- Using methods of theoretical computer science.
- High expressive power and hard recognition
- Any number-theoretic methods?

Conclusion

- A basic mathematical object.
- Using methods of theoretical computer science.
- High expressive power and hard recognition
- Any number-theoretic methods?

Problem

Construct a set not representable by equations with $\{\cup, \cap, \boxplus\}$.