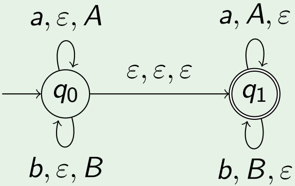
Of stacks (of stacks (...) with blind counters) with blind counters

Georg Zetzsche

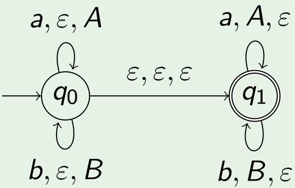
Technische Universität Kaiserslautern

Algorithmics on Infinite State Systems 2014

Example (Pushdown automaton)

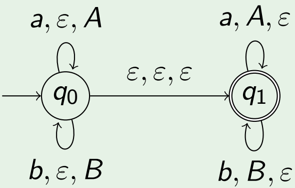


Example (Pushdown automaton)



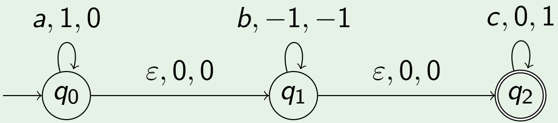
$$L = \{ww^{\mathrm{rev}} \mid w \in \{a, b\}^*\}$$

Example (Pushdown automaton)

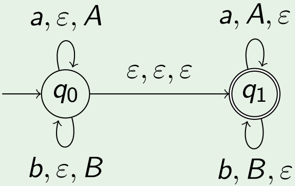


$$L = \{ww^{\mathrm{rev}} \mid w \in \{a, b\}^*\}$$

Example (Blind counter automaton)

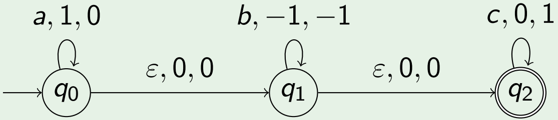


Example (Pushdown automaton)



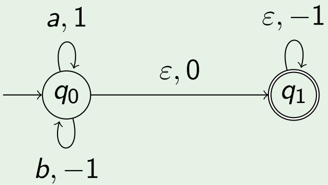
$$L = \{ww^{\text{rev}} \mid w \in \{a, b\}^*\}$$

Example (Blind counter automaton)

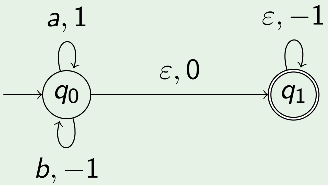


$$L = \{a^n b^n c^n \mid n \geqslant 0\}$$

Example (Partially blind counter automaton)



Example (Partially blind counter automaton)



$$L = \{w \in \{a, b\}^* \quad | \quad |p|_a \geqslant |p|_b \text{ for each prefix } p \text{ of } w\}$$

Automata models that extend finite automata by some storage mechanism:

- Pushdown automata
- Blind counter automata
- Partially blind counter automata
- Turing machines

Automata models that extend finite automata by some storage mechanism:

- Pushdown automata
- Blind counter automata
- Partially blind counter automata
- Turing machines

Each storage mechanism consists of:

- States: set S of states
- Operations: partial maps $\alpha_1, \ldots, \alpha_n \colon S \to S$

Model	States	Operations
Pushdown automata	$S = \Gamma^*$	$\mathrm{push}_a\colon w \mapsto wa,\ a \in \Gamma$ $\mathrm{pop}_a\colon wa \mapsto w,\ a \in \Gamma$

Model	States	Operations
Pushdown automata	$S = \Gamma^*$	$\mathrm{push}_a \colon w \mapsto wa,\ a \in \Gamma$ $\mathrm{pop}_a \colon wa \mapsto w,\ a \in \Gamma$
Blind counter automata	$S = \mathbb{Z}^n$	$\mathrm{inc}_i \colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i + 1, \ldots, x_n)$ $\mathrm{dec}_i \colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i - 1, \ldots, x_n)$

Model	States	Operations
Pushdown automata	$S = \Gamma^*$	$\mathrm{push}_a \colon w \mapsto wa,\ a \in \Gamma$ $\mathrm{pop}_a \colon wa \mapsto w,\ a \in \Gamma$
Blind counter automata	$S = \mathbb{Z}^n$	$\mathrm{inc}_i \colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i + 1, \ldots, x_n)$ $\mathrm{dec}_i \colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i - 1, \ldots, x_n)$
Partially blind counter automata	$S = \mathbb{N}^n$	$\mathrm{inc}_i \colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i + 1, \ldots, x_n)$ $\mathrm{dec}_i \colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i - 1, \ldots, x_n)$

Model	States	Operations
Pushdown automata	$S = \Gamma^*$	$\mathrm{push}_a\colon w \mapsto wa,\ a \in \Gamma$ $\mathrm{pop}_a\colon wa \mapsto w,\ a \in \Gamma$
Blind counter automata	$S = \mathbb{Z}^n$	$\mathrm{inc}_i\colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i + 1, \ldots, x_n)$ $\mathrm{dec}_i\colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i - 1, \ldots, x_n)$
Partially blind counter automata	$S = \mathbb{N}^n$	$\mathrm{inc}_i\colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i + 1, \ldots, x_n)$ $\mathrm{dec}_i\colon (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_i - 1, \ldots, x_n)$

Observation

Here, a sequence $\beta_1, \ldots, \beta_k$ of operations is valid if and only if

$$\beta_1 \circ \cdots \circ \beta_k = \mathrm{id}$$

Definition

A *monoid* is

- a set M together with
- an associative binary operation $\cdot\colon M \times M \to M$ and
- a neutral element $1 \in M$ ($a1 = 1a = a$ for any $a \in M$).

Definition

A *monoid* is

- a set M together with
- an associative binary operation $\cdot \colon M \times M \to M$ and
- a neutral element $1 \in M$ ($a1 = 1a = a$ for any $a \in M$).

Storage mechanisms as monoids

- Let S be a set of states and $\alpha_1, \ldots, \alpha_n \colon S \to S$ partial maps.
- The set of all compositions of $\alpha_1, \ldots, \alpha_n$ is a monoid M.
- The identity map is the neutral element of M.
- M is a decription of the storage mechanism.

Valence automata

Common generalization: Valence Automata

Valence automaton over M:

- Finite automaton with edges $p \xrightarrow{w|m} q$, $w \in \Sigma^*$, $m \in M$.

Valence automata

Common generalization: Valence Automata

Valence automaton over M:

- Finite automaton with edges $p \xrightarrow{w|m} q$, $w \in \Sigma^*$, $m \in M$.
- Run $q_0 \xrightarrow{w_1|m_1} q_1 \xrightarrow{w_2|m_2} \cdots \xrightarrow{w_n|m_n} q_n$ is *accepting for* $w_1 \cdots w_n$ if
  - q_0 is the initial state,
  - q_n is a final state, and

Valence automata

Common generalization: Valence Automata

Valence automaton over M:

- Finite automaton with edges $p \xrightarrow{w|m} q$, $w \in \Sigma^*$, $m \in M$.
- Run $q_0 \xrightarrow{w_1|m_1} q_1 \xrightarrow{w_2|m_2} \cdots \xrightarrow{w_n|m_n} q_n$ is *accepting for* $w_1 \cdots w_n$ if
  - q_0 is the initial state,
  - q_n is a final state, and
  - $m_1 \cdots m_n = 1$.

Valence automata

Common generalization: Valence Automata

Valence automaton over M:

- Finite automaton with edges $p \xrightarrow{w|m} q$, $w \in \Sigma^*$, $m \in M$.
- Run $q_0 \xrightarrow{w_1|m_1} q_1 \xrightarrow{w_2|m_2} \cdots \xrightarrow{w_n|m_n} q_n$ is *accepting for* $w_1 \cdots w_n$ if
  - q_0 is the initial state,
  - q_n is a final state, and
  - $m_1 \cdots m_n = 1$.

Language class

VA(M) languages accepted by valence automata over M.

Classical results can now be generalized:

Questions

- For which storage mechanisms can we avoid silent transitions?

Classical results can now be generalized:

Questions

- For which storage mechanisms can we avoid silent transitions?
- For which do we have semilinearity of all languages?

Classical results can now be generalized:

Questions

- For which storage mechanisms can we avoid silent transitions?
- For which do we have semilinearity of all languages?
- For which is the language class, for example, Boolean closed?

Classical results can now be generalized:

Questions

- For which storage mechanisms can we avoid silent transitions?
- For which do we have semilinearity of all languages?
- For which is the language class, for example, Boolean closed?
- For which can we decide, for example, emptiness?

Monoids defined by graphs

By graphs, we mean undirected graphs with loops allowed.

Monoids defined by graphs

By graphs, we mean undirected graphs with loops allowed.

Let $\Gamma = (V, E)$ be a graph. Let

$$X_\Gamma = \{a_v, \bar{a}_v \mid v \in V\}$$

Monoids defined by graphs

By graphs, we mean undirected graphs with loops allowed.
Let $\Gamma = (V, E)$ be a graph. Let

$$X_\Gamma = \{a_v, \bar{a}_v \mid v \in V\}$$
$$R_\Gamma = \{a_v \bar{a}_v = \varepsilon \mid v \in V\}$$

Monoids defined by graphs

By graphs, we mean undirected graphs with loops allowed.
Let $\Gamma = (V, E)$ be a graph. Let

$$\begin{aligned} X_\Gamma &= \{a_v, \bar{a}_v \mid v \in V\} \\ R_\Gamma &= \{a_v \bar{a}_v = \varepsilon \mid v \in V\} \\ &\quad \cup \{xy = yx \mid x \in \{a_u, \bar{a}_u\}, y \in \{a_v, \bar{a}_v\}, \{u, v\} \in E\} \end{aligned}$$

Monoids defined by graphs

By graphs, we mean undirected graphs with loops allowed.
Let $\Gamma = (V, E)$ be a graph. Let

$$X_\Gamma = \{a_v, \bar{a}_v \mid v \in V\}$$
$$R_\Gamma = \{a_v \bar{a}_v = \varepsilon \mid v \in V\}$$
$$\cup \{xy = yx \mid x \in \{a_u, \bar{a}_u\}, y \in \{a_v, \bar{a}_v\}, \{u, v\} \in E\}$$

$$\mathbb{M}\Gamma = X_\Gamma^* / R_\Gamma$$

Monoids defined by graphs

By graphs, we mean undirected graphs with loops allowed.
Let $\Gamma = (V, E)$ be a graph. Let

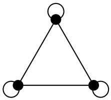
$$\begin{aligned} X_\Gamma &= \{a_v, \bar{a}_v \mid v \in V\} \\ R_\Gamma &= \{a_v\bar{a}_v = \varepsilon \mid v \in V\} \\ &\quad \cup \{xy = yx \mid x \in \{a_u, \bar{a}_u\}, y \in \{a_v, \bar{a}_v\}, \{u, v\} \in E\} \end{aligned}$$

$$\mathbb{M}\Gamma = X_\Gamma^*/R_\Gamma$$

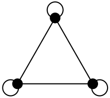
Intuition

- $\mathbb{B}$: bicyclic monoid, $\mathbb{B} = \{a, \bar{a}\}^*/\{a\bar{a} = \varepsilon\}$.
- $\mathbb{Z}$: group of integers
- For each unlooped vertex, we have a copy of $\mathbb{B}$
- For each looped vertex, we have a copy of $\mathbb{Z}$
- $\mathbb{M}\Gamma$ consists of sequences of such elements
- An edge between vertices means that elements can commute

Examples

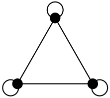


Examples



$\mathbb{Z}^3$

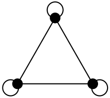
Examples



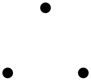
$\mathbb{Z}^3$

Blind counter

Examples

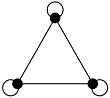


$\mathbb{Z}^3$

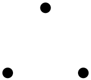


Blind counter

Examples



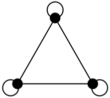
$\mathbb{Z}^3$



$\mathbb{B} * \mathbb{B} * \mathbb{B}$

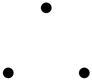
Blind counter

Examples



$\mathbb{Z}^3$

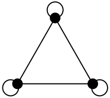
Blind counter



$\mathbb{B} * \mathbb{B} * \mathbb{B}$

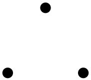
Pushdown

Examples



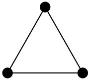
$\mathbb{Z}^3$

Blind counter

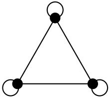


$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

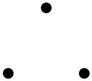


Examples



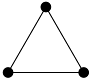
$\mathbb{Z}^3$

Blind counter



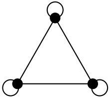
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



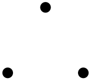
$\mathbb{B}^3$

Examples



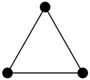
$\mathbb{Z}^3$

Blind counter



$\mathbb{B} * \mathbb{B} * \mathbb{B}$

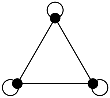
Pushdown



$\mathbb{B}^3$

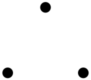
Partially blind counter

Examples



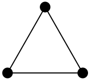
$\mathbb{Z}^3$

Blind counter



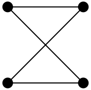
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

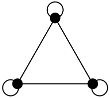


$\mathbb{B}^3$

Partially blind counter

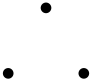


Examples



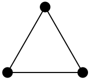
$\mathbb{Z}^3$

Blind counter



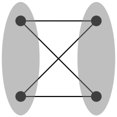
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

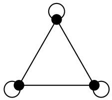


$\mathbb{B}^3$

Partially blind counter

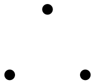


Examples



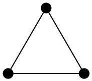
$\mathbb{Z}^3$

Blind counter



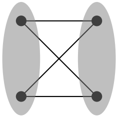
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



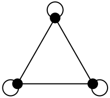
$\mathbb{B}^3$

Partially blind counter



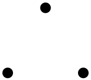
$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

Examples



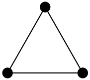
$\mathbb{Z}^3$

Blind counter



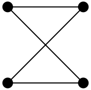
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



$\mathbb{B}^3$

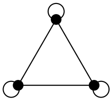
Partially blind counter



$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

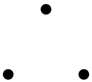
Infinite tape (TM)

Examples



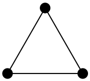
$\mathbb{Z}^3$

Blind counter



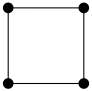
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



$\mathbb{B}^3$

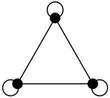
Partially blind counter



$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

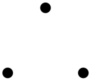
Infinite tape (TM)

Examples



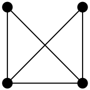
$\mathbb{Z}^3$

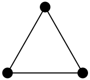
Blind counter



$\mathbb{B} * \mathbb{B} * \mathbb{B}$

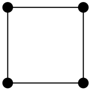
Pushdown





$\mathbb{B}^3$

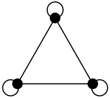
Partially blind counter



$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

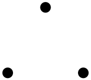
Infinite tape (TM)

Examples



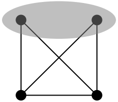
$\mathbb{Z}^3$

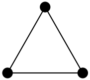
Blind counter



$\mathbb{B} * \mathbb{B} * \mathbb{B}$

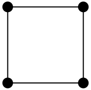
Pushdown





$\mathbb{B}^3$

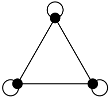
Partially blind counter



$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

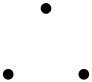
Infinite tape (TM)

Examples



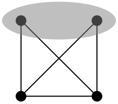
$\mathbb{Z}^3$

Blind counter

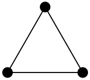


$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

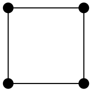


$(\mathbb{B} * \mathbb{B}) \times \mathbb{B} \times \mathbb{B}$



$\mathbb{B}^3$

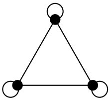
Partially blind counter



$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

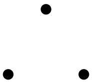
Infinite tape (TM)

Examples



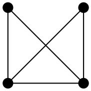
$\mathbb{Z}^3$

Blind counter



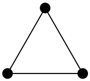
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



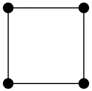
$(\mathbb{B} * \mathbb{B}) \times \mathbb{B} \times \mathbb{B}$

Pushdown + partially blind counters



$\mathbb{B}^3$

Partially blind counter



$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

Infinite tape (TM)

Silent Transitions

A transition that reads no input is called *silent transition* or ε-*transition*.

Silent Transitions

A transition that reads no input is called *silent transition* or ε-*transition*.

Important problem

- When can silent transitions be eliminated?
- Without silent transitions, membership in NP.
- Elimination can be regarded as a precomputation.

Silent Transitions

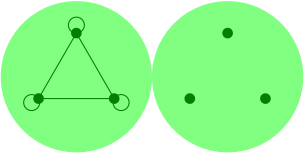
A transition that reads no input is called *silent transition* or ε-*transition*.

Important problem

- When can silent transitions be eliminated?
- Without silent transitions, membership in NP.
- Elimination can be regarded as a precomputation.

Question

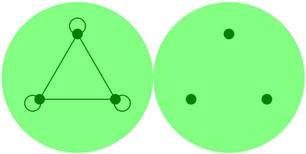
For which storage mechanisms can we avoid silent transitions?

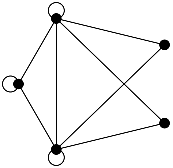


Theorem (Z., ICALP 2013)

Let Γ be a graph such that

- *any two looped vertices are adjacent,*
- *no two unlooped vertices are adjacent.*

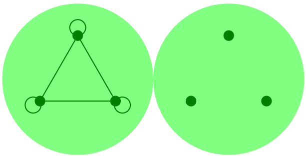


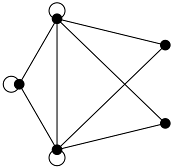


Theorem (Z., ICALP 2013)

Let Γ be a graph such that

- *any two looped vertices are adjacent,*
- *no two unlooped vertices are adjacent.*





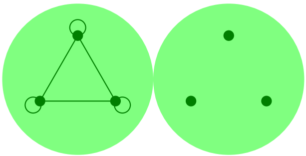
Theorem (Z., ICALP 2013)

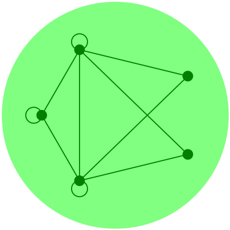
Let Γ *be a graph such that*

- *any two looped vertices are adjacent,*
- *no two unlooped vertices are adjacent.*

Then the following conditions are equivalent:

- *Silent transitions can be avoided over* $\mathbb{M}\Gamma$*.*
- Γ *does not contain*  *as an induced subgraph.*





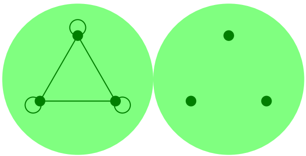
Theorem (Z., ICALP 2013)

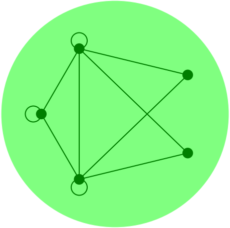
Let Γ be a graph such that

- *any two looped vertices are adjacent,*
- *no two unlooped vertices are adjacent.*

Then the following conditions are equivalent:

- *Silent transitions can be avoided over $\mathbb{M}\Gamma$.*
- *Γ does not contain*  *as an induced subgraph.*





Theorem (Z., ICALP 2013)

Let Γ be a graph such that

- *any two looped vertices are adjacent,*
- *no two unlooped vertices are adjacent.*

Then the following conditions are equivalent:

- *Silent transitions can be avoided over* $\mathbb{M}\Gamma$.
- Γ *does not contain*  *as an induced subgraph.*
- $\mathbb{M}\Gamma \in \mathsf{StCtr}$

Positive case

Definition (Stacked counters)

Let StCtr be the smallest class of monoids such that

- $1 \in \mathsf{StCtr}$
- if $M \in \mathsf{StCtr}$, then $M \times \mathbb{Z} \in \mathsf{StCtr}$
- if $M \in \mathsf{StCtr}$, then $M * \mathbb{B} \in \mathsf{StCtr}$

Positive case

Definition (Stacked counters)

Let StCtr be the smallest class of monoids such that

- $1 \in \mathsf{StCtr}$
- if $M \in \mathsf{StCtr}$, then $M \times \mathbb{Z} \in \mathsf{StCtr}$
- if $M \in \mathsf{StCtr}$, then $M * \mathbb{B} \in \mathsf{StCtr}$

Interpretation of StCtr

StCtr corresponds to the class of storage mechanisms obtained by

- adding a blind counter ($M \times \mathbb{Z}$):
  - States: (c, z), c an old state, $z \in \mathbb{Z}$.
  - Operations: old operations; increment, decrement for counter

Positive case

Definition (Stacked counters)

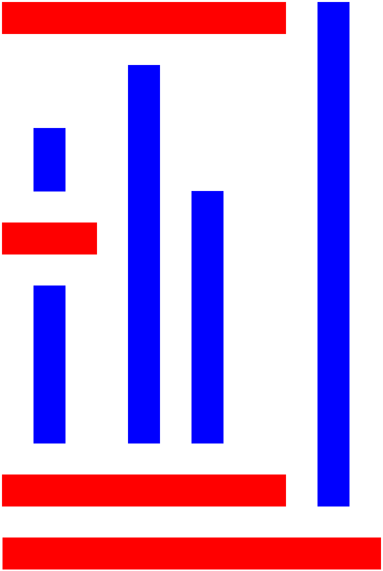
Let StCtr be the smallest class of monoids such that

- $1 \in \mathsf{StCtr}$
- if $M \in \mathsf{StCtr}$, then $M \times \mathbb{Z} \in \mathsf{StCtr}$
- if $M \in \mathsf{StCtr}$, then $M * \mathbb{B} \in \mathsf{StCtr}$

Interpretation of StCtr

StCtr corresponds to the class of storage mechanisms obtained by

- adding a blind counter ($M \times \mathbb{Z}$):
  - States: (c, z), c an old state, $z \in \mathbb{Z}$.
  - Operations: old operations; increment, decrement for counter
- building stacks ($M * \mathbb{B}$)
  - States: sequences $\square c_1 \square c_2 \square \cdots \square c_n$, c_i old states
  - Operations: push separator, pop if empty, manipulate topmost entry



Semilinearity

For which monoids M are all languages in $\mathsf{VA}(M)$ semilinear?

- Parikh's Theorem: Pushdown automata
- Ibarra + Greibach: Blind counter automata

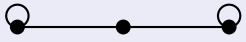
Semilinearity

For which monoids M are all languages in $\mathsf{VA}(M)$ semilinear?

- Parikh's Theorem: Pushdown automata
- Ibarra + Greibach: Blind counter automata

Theorem (Buckheister, Z., MFCS 2013)

Let Γ be a graph. The following conditions are equivalent:

- *All languages in* $\mathsf{VA}(\mathbb{M}\Gamma)$ *are semilinear.*
- Γ *satisfies:*
  1. Γ *contains neither*  *nor*  *as an induced subgraph and*
  2. Γ*, minus loops, is a transitive forest.*

Semilinearity

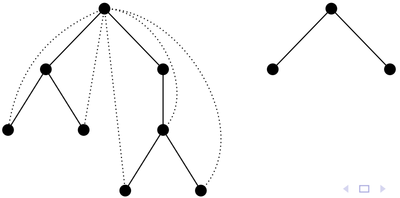
For which monoids M are all languages in $\mathsf{VA}(M)$ semilinear?

- Parikh's Theorem: Pushdown automata
- Ibarra + Greibach: Blind counter automata

Theorem (Buckheister, Z., MFCS 2013)

Let Γ be a graph. The following conditions are equivalent:

- *All languages in $\mathsf{VA}(\mathbb{M}\Gamma)$ are semilinear.*
- *Γ satisfies:*
  1. *Γ contains neither nor as an induced subgraph and*
  2. *Γ, minus loops, is a transitive forest.*



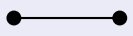
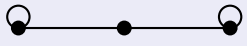
Semilinearity

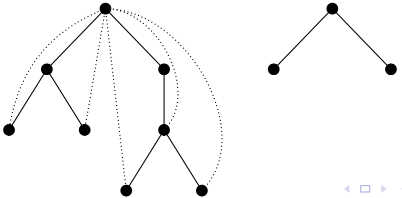
For which monoids M are all languages in $\mathsf{VA}(M)$ semilinear?

- Parikh's Theorem: Pushdown automata
- Ibarra + Greibach: Blind counter automata

Theorem (Buckheister, Z., MFCS 2013)

Let Γ be a graph. The following conditions are equivalent:

- *All languages in* $\mathsf{VA}(\mathbb{M}\Gamma)$ *are semilinear.*
- Γ *satisfies:*
  1. Γ *contains neither*  *nor*  *as an induced subgraph and*
  2. Γ*, minus loops, is a transitive forest.*
- $\mathsf{VA}(\mathbb{M}\Gamma) \subseteq \mathsf{VA}(M)$ *for some* $M \in \mathsf{StCtr}$*. (NP-membership!)*



Expressiveness

Algebraic extensions

Let $\mathcal{F}$ be a language class. An $\mathcal{F}$*-grammar* G consists of

- Nonterminals N, terminals T, start symbol $S \in N$
- Productions $A \to L$ with $L \subseteq (N \cup T)^*$, $L \in \mathcal{F}$

Expressiveness

Algebraic extensions

Let $\mathcal{F}$ be a language class. An *$\mathcal{F}$-grammar* G consists of

- Nonterminals N, terminals T, start symbol $S \in N$
- Productions $A \to L$ with $L \subseteq (N \cup T)^*$, $L \in \mathcal{F}$

$$uAv \Rightarrow uwv \quad \text{whenever } w \in L.$$

Expressiveness

Algebraic extensions

Let $\mathcal{F}$ be a language class. An *$\mathcal{F}$-grammar G* consists of

- Nonterminals N, terminals T, start symbol $S \in N$
- Productions $A \to L$ with $L \subseteq (N \cup T)^*$, $L \in \mathcal{F}$

$$uAv \Rightarrow uwv \quad \text{whenever } w \in L.$$

- Generated language: $\{w \in T^* \mid S \Rightarrow^* w\}$.

Expressiveness

Algebraic extensions

Let $\mathcal{F}$ be a language class. An *$\mathcal{F}$-grammar G* consists of

- Nonterminals N, terminals T, start symbol $S \in N$
- Productions $A \to L$ with $L \subseteq (N \cup T)^*$, $L \in \mathcal{F}$
$$uAv \Rightarrow uwv \quad \text{whenever } w \in L.$$
- Generated language: $\{w \in T^* \mid S \Rightarrow^* w\}$.
- Such languages are *algebraic over $\mathcal{F}$*, class denoted $\mathrm{Alg}(\mathcal{F})$.

Expressiveness

Algebraic extensions

Let $\mathcal{F}$ be a language class. An *$\mathcal{F}$-grammar G* consists of

- Nonterminals N, terminals T, start symbol $S \in N$
- Productions $A \to L$ with $L \subseteq (N \cup T)^*$, $L \in \mathcal{F}$

$$uAv \Rightarrow uwv \quad \text{whenever } w \in L.$$

- Generated language: $\{w \in T^* \mid S \Rightarrow^* w\}$.
- Such languages are *algebraic over $\mathcal{F}$*, class denoted $\mathrm{Alg}(\mathcal{F})$.

Presburger constraints

For each language class $\mathcal{F}$, $\mathrm{SLI}(\mathcal{F})$ denotes the class of languages

$$h(L \cap \Psi^{-1}(S))$$

for some $L \in \mathcal{F}$, a homomorphism h and a semilinear set S.

A hierarchy of language classes

Hierarchy

$$\mathsf{F}_0 = \text{finite languages},$$

$$\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i), \qquad \mathsf{F}_{i+1} = \mathsf{SLI}(\mathsf{G}_i), \qquad \mathsf{F} = \bigcup_{i \geqslant 0} \mathsf{F}_i.$$

A hierarchy of language classes

Hierarchy

$$F_0 = \text{finite languages},$$

$$G_i = \mathrm{Alg}(F_i), \qquad F_{i+1} = \mathrm{SLI}(G_i), \qquad F = \bigcup_{i \geqslant 0} F_i.$$

In particular: $G_0 = \mathrm{CF}$.

A hierarchy of language classes

Hierarchy

$$F_0 = \text{finite languages},$$

$$G_i = \mathsf{Alg}(F_i), \qquad F_{i+1} = \mathsf{SLI}(G_i), \qquad F = \bigcup_{i \geqslant 0} F_i.$$

In particular: $G_0 = \mathsf{CF}$.

$$F_0 \subseteq G_0 \subseteq F_1 \subseteq G_1 \subseteq \cdots \subseteq F$$

A hierarchy of language classes

Hierarchy

$$F_0 = \text{finite languages},$$

$$G_i = \mathsf{Alg}(F_i), \qquad F_{i+1} = \mathsf{SLI}(G_i), \qquad F = \bigcup_{i \geqslant 0} F_i.$$

In particular: $G_0 = \mathsf{CF}$.

$$F_0 \subseteq G_0 \subseteq F_1 \subseteq G_1 \subseteq \cdots \subseteq F$$

Theorem

$\mathsf{VA}(\mathbb{B} * \mathbb{B} * M) = \mathsf{Alg}(\mathsf{VA}(M))$

A hierarchy of language classes

Hierarchy

$$\mathsf{F}_0 = \text{finite languages},$$

$$\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i), \qquad \mathsf{F}_{i+1} = \mathsf{SLI}(\mathsf{G}_i), \qquad \mathsf{F} = \bigcup_{i \geqslant 0} \mathsf{F}_i.$$

In particular: $\mathsf{G}_0 = \mathsf{CF}$.

$$\mathsf{F}_0 \subseteq \mathsf{G}_0 \subseteq \mathsf{F}_1 \subseteq \mathsf{G}_1 \subseteq \cdots \subseteq \mathsf{F}$$

Theorem

$\mathsf{VA}(\mathbb{B} * \mathbb{B} * M) = \mathsf{Alg}(\mathsf{VA}(M))$, $\bigcup_{i \geqslant 0} \mathsf{VA}(M \times \mathbb{Z}^i) = \mathsf{SLI}(\mathsf{VA}(M))$.

A hierarchy of language classes

Hierarchy

$$F_0 = \text{finite languages},$$

$$G_i = \mathrm{Alg}(F_i), \qquad F_{i+1} = \mathrm{SLI}(G_i), \qquad F = \bigcup_{i \geqslant 0} F_i.$$

In particular: $G_0 = \mathrm{CF}$.

$$F_0 \subseteq G_0 \subseteq F_1 \subseteq G_1 \subseteq \cdots \subseteq F$$

Theorem

$\mathrm{VA}(\mathbb{B} * \mathbb{B} * M) = \mathrm{Alg}(\mathrm{VA}(M))$, $\bigcup_{i \geqslant 0} \mathrm{VA}(M \times \mathbb{Z}^i) = \mathrm{SLI}(\mathrm{VA}(M))$.

Corollary

Stacked counter automata accept precisely the languages in F.

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Applications

- $L{\downarrow}$ is observed through a lossy channel.

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Applications

- $L{\downarrow}$ is observed through a lossy channel. Decidability for REG!

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Applications

- $L{\downarrow}$ is observed through a lossy channel. Decidability for REG!
- Decide reversal boundedness. $\triangle$: up, $\triangledown$: down; $(\triangle\triangledown)^* \subseteq L{\downarrow}$?

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Applications

- $L{\downarrow}$ is observed through a lossy channel. Decidability for REG!
- Decide reversal boundedness. $\triangle$: up, $\triangledown$: down; $(\triangle\triangledown)^* \subseteq L{\downarrow}$?

Computability

For which systems can we compute $L{\downarrow}$?

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Applications

- $L{\downarrow}$ is observed through a lossy channel. Decidability for REG!
- Decide reversal boundedness. $\vartriangle$: up, $\triangledown$: down; $(\vartriangle\triangledown)^* \subseteq L{\downarrow}$?

Computability

For which systems can we compute $L{\downarrow}$?

- for $\mathsf{Alg}(\mathcal{F})$ whenever computable for $\mathcal{F}$ (van Leeuwen 1978)

Downward closures

$u \preceq v$: u is obtained from v by arbitrarily deleting symbols

Theorem (Higman)

For every language $L \subseteq X^*$, the set $L{\downarrow} = \{u \in X^* \mid u \preceq v \text{ for some } v \in L\}$ is regular.

Applications

- $L{\downarrow}$ is observed through a lossy channel. Decidability for REG!
- Decide reversal boundedness. $\triangle$: up, $\triangledown$: down; $(\triangle\triangledown)^* \subseteq L{\downarrow}$?

Computability

For which systems can we compute $L{\downarrow}$?

- for $\mathsf{Alg}(\mathcal{F})$ whenever computable for $\mathcal{F}$ (van Leeuwen 1978)
- for Petri net languages (Habermehl, Meyer, Wimmel, ICALP 2010)

Computing the downward closure

Theorem

For stacked counter automata, downward closures can be computed.

Computing the downward closure

Theorem

For stacked counter automata, downward closures can be computed.

Problem

- Computability preserved by $\mathsf{Alg}(\cdot)$

Computing the downward closure

Theorem

For stacked counter automata, downward closures can be computed.

Problem

- Computability preserved by $\mathsf{Alg}(\cdot)$
- Preservation not clear for $\mathsf{SLI}(\cdot)$ (probably not true)

Computing the downward closure

Theorem

For stacked counter automata, downward closures can be computed.

Problem

- Computability preserved by $\mathsf{Alg}(\cdot)$
- Preservation not clear for $\mathsf{SLI}(\cdot)$ (probably not true)
- Hence: Stronger invariant

Computing the downward closure

Theorem

For stacked counter automata, downward closures can be computed.

Problem

- Computability preserved by $\mathsf{Alg}(\cdot)$
- Preservation not clear for $\mathsf{SLI}(\cdot)$ (probably not true)
- Hence: Stronger invariant

Parikh annotations

- New language in the same class
- Additional symbols encode decomposition of Parikh image into constant and period vectors
- Adding period vectors by inserting at designated positions

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a+b)^{\oplus} + a^{\oplus}\big) \cup \big(d + (a+b)^{\oplus} + b^{\oplus}\big)$.

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a+b)^{\oplus} + a^{\oplus}\big) \cup \big(d + (a+b)^{\oplus} + b^{\oplus}\big)$.

$$\begin{aligned} P &= \{p, q, r, s\}, \\ C &= \{e, f\}, \\ P_e &= \{p, q\}, \\ P_f &= \{r, s\}, \end{aligned}$$

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\left(c + (a+b)^{\oplus} + a^{\oplus}\right) \cup \left(d + (a+b)^{\oplus} + b^{\oplus}\right)$.

$$\begin{aligned}
P &= \{p, q, r, s\}, & & & & \\
C &= \{e, f\}, & \varphi(e) &= c, & \varphi(f) &= d, \\
P_e &= \{p, q\}, & \varphi(p) &= a + b, & \varphi(q) &= a, \\
P_f &= \{r, s\}, & \varphi(r) &= a + b, & \varphi(s) &= b,
\end{aligned}$$

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a + b)^{\oplus} + a^{\oplus}\big) \cup \big(d + (a + b)^{\oplus} + b^{\oplus}\big)$.

$$\begin{aligned} P &= \{p, q, r, s\}, & & & & \\ C &= \{e, f\}, & \varphi(e) &= c, & \varphi(f) &= d, \\ P_e &= \{p, q\}, & \varphi(p) &= a + b, & \varphi(q) &= a, \\ P_f &= \{r, s\}, & \varphi(r) &= a + b, & \varphi(s) &= b, \end{aligned}$$

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a+b)^{\oplus} + a^{\oplus}\big) \cup \big(d + (a+b)^{\oplus} + b^{\oplus}\big)$.

$$\begin{aligned}
P &= \{p, q, r, s\}, & & \\
C &= \{e, f\}, & \varphi(e) &= c, & \varphi(f) &= d, \\
P_e &= \{p, q\}, & \varphi(p) &= a+b, & \varphi(q) &= a, \\
P_f &= \{r, s\}, & \varphi(r) &= a+b, & \varphi(s) &= b,
\end{aligned}$$

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a+b)^{\oplus} + a^{\oplus}\big) \cup \big(d + (a+b)^{\oplus} + b^{\oplus}\big)$.

$$\begin{aligned}
P &= \{p, q, r, s\}, & & & & \\
C &= \{e, f\}, & \varphi(e) &= c, & \varphi(f) &= d, \\
P_e &= \{p, q\}, & \varphi(p) &= a + b, & \varphi(q) &= a, \\
P_f &= \{r, s\}, & \varphi(r) &= a + b, & \varphi(s) &= b,
\end{aligned}$$

$$K = e \diamond (pab)^* c \diamond (qa)^* \quad \cup \quad f \diamond (rab)^* d \diamond (sb)^*$$

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a+b)^\oplus + a^\oplus\big) \cup \big(d + (a+b)^\oplus + b^\oplus\big)$.

$$\begin{aligned}
P &= \{p, q, r, s\}, && && \\
C &= \{e, f\}, && \varphi(e) = c, && \varphi(f) = d, \\
P_e &= \{p, q\}, && \varphi(p) = a + b, && \varphi(q) = a, \\
P_f &= \{r, s\}, && \varphi(r) = a + b, && \varphi(s) = b,
\end{aligned}$$

$$K = e \diamond (pab)^* c \diamond (qa)^* \quad \cup \quad f \diamond (rab)^* d \diamond (sb)^*$$

- Makes Parikh decomposition accessible to transducers

Parikh annotations

Example

$$L = (ab)^*(ca^* \cup db^*)$$

Parikh image: $\big(c + (a+b)^\oplus + a^\oplus\big) \cup \big(d + (a+b)^\oplus + b^\oplus\big)$.

$$\begin{aligned}
P &= \{p, q, r, s\}, & & & & \\
C &= \{e, f\}, & \varphi(e) &= c, & \varphi(f) &= d, \\
P_e &= \{p, q\}, & \varphi(p) &= a+b, & \varphi(q) &= a, \\
P_f &= \{r, s\}, & \varphi(r) &= a+b, & \varphi(s) &= b,
\end{aligned}$$

$$K = e \diamond (pab)^* c \diamond (qa)^* \quad \cup \quad f \diamond (rab)^* d \diamond (sb)^*$$

- Makes Parikh decomposition accessible to transducers
- Pumping lemma described by a language

Theorem

For each level F_i*, one can compute Parikh annotations in* F_i*.*

Theorem

For each level F_i*, one can compute Parikh annotations in* F_i*.*

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm

Theorem

For each level F_i, one can compute Parikh annotations in F_i.

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm
- For $L \in \mathsf{F}_i = \mathsf{SLI}(\mathsf{G}_{i-1})$

Theorem

For each level F_i, one can compute Parikh annotations in F_i.

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm
- For $L \in \mathsf{F}_i = \mathsf{SLI}(\mathsf{G}_{i-1})$, write $L = h(L' \cap \Psi^{-1}(S))$, $L' \in \mathsf{G}_{i-1}$

Theorem

For each level F_i*, one can compute Parikh annotations in* F_i*.*

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm
- For $L \in \mathsf{F}_i = \mathsf{SLI}(\mathsf{G}_{i-1})$, write $L = h(L' \cap \Psi^{-1}(S))$, $L' \in \mathsf{G}_{i-1}$
- Construct Parikh annotation $A \in \mathsf{G}_{i-1}$ for L'

Theorem

For each level F_i*, one can compute Parikh annotations in* F_i*.*

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm
- For $L \in \mathsf{F}_i = \mathsf{SLI}(\mathsf{G}_{i-1})$, write $L = h(L' \cap \Psi^{-1}(S))$, $L' \in \mathsf{G}_{i-1}$
- Construct Parikh annotation $A \in \mathsf{G}_{i-1}$ for L'
- From A, compute $M \in \mathsf{G}_{i-1}$ with $L \subseteq M \subseteq L{\downarrow}$

Theorem

For each level F_i*, one can compute Parikh annotations in* F_i*.*

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm
- For $L \in \mathsf{F}_i = \mathsf{SLI}(\mathsf{G}_{i-1})$, write $L = h(L' \cap \Psi^{-1}(S))$, $L' \in \mathsf{G}_{i-1}$
- Construct Parikh annotation $A \in \mathsf{G}_{i-1}$ for L'
- From A, compute $M \in \mathsf{G}_{i-1}$ with $L \subseteq M \subseteq L{\downarrow}$, hence $M{\downarrow} = L{\downarrow}$.

Theorem

For each level F_i*, one can compute Parikh annotations in* F_i*.*

Computing downward closures

Recursively with respect to the hierarchy level:

- For $\mathsf{G}_i = \mathsf{Alg}(\mathsf{F}_i)$, use van Leeuwen's algorithm
- For $L \in \mathsf{F}_i = \mathsf{SLI}(\mathsf{G}_{i-1})$, write $L = h(L' \cap \Psi^{-1}(S))$, $L' \in \mathsf{G}_{i-1}$
- Construct Parikh annotation $A \in \mathsf{G}_{i-1}$ for L'
- From A, compute $M \in \mathsf{G}_{i-1}$ with $L \subseteq M \subseteq L\!\downarrow$, hence $M\!\downarrow = L\!\downarrow$.

Other applications of Parikh annotations include:

Theorem

For each $i \geqslant 0$*:* $\mathsf{F}_i \subsetneq \mathsf{G}_i \subsetneq \mathsf{F}_{i+1}$*.*

Conclusion

- Silent transitions avoidable, non-uniform membership in NP

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds
- Downward closure computable

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds
- Downward closure computable
- Strict hierarchy of language classes

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds
- Downward closure computable
- Strict hierarchy of language classes

More classical results can be generalized:

Ongoing work

- Uniform word problem, connections to group theory

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds
- Downward closure computable
- Strict hierarchy of language classes

More classical results can be generalized:

Ongoing work

- Uniform word problem, connections to group theory
- Decidability of logics over reachability graphs

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds
- Downward closure computable
- Strict hierarchy of language classes

More classical results can be generalized:

Ongoing work

- Uniform word problem, connections to group theory
- Decidability of logics over reachability graphs
- pre*/post* computation

Conclusion

- Silent transitions avoidable, non-uniform membership in NP
- Parikh's Theorem holds
- Downward closure computable
- Strict hierarchy of language classes

More classical results can be generalized:

Ongoing work

- Uniform word problem, connections to group theory
- Decidability of logics over reachability graphs
- pre*/post* computation
- Decidability of questions for Büchi variants