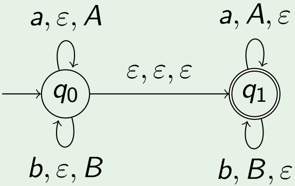
The Emptiness Problem for Valence Automata or: Another Decidable Extension of Petri Nets

Georg Zetzsche

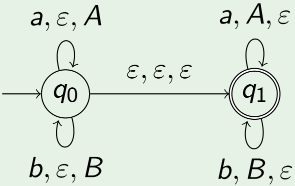
Technische Universität Kaiserslautern

Reachability Problems 2015

Example (Pushdown automaton)

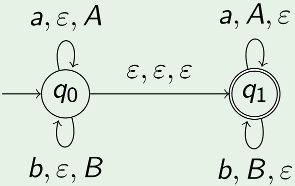


Example (Pushdown automaton)



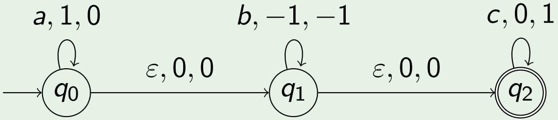
$$L = \{ww^{\text{rev}} \mid w \in \{a, b\}^*\}$$

Example (Pushdown automaton)

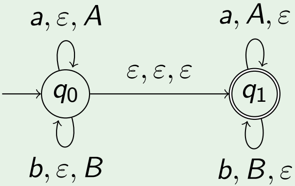


$$L = \{ww^{\text{rev}} \mid w \in \{a, b\}^*\}$$

Example (Blind counter automaton)

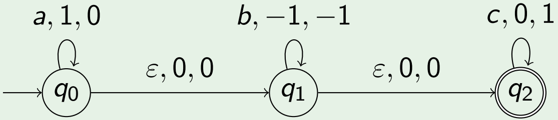


Example (Pushdown automaton)



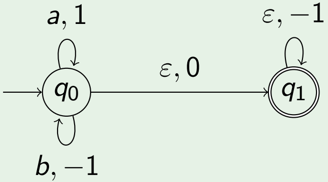
$$L = \{ww^{\text{rev}} \mid w \in \{a, b\}^*\}$$

Example (Blind counter automaton)

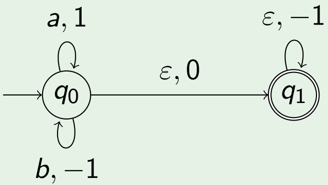


$$L = \{a^n b^n c^n \mid n \geqslant 0\}$$

Example (Partially blind counter automaton)



Example (Partially blind counter automaton)



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

Storage mechanisms

Automata models that extend finite automata by some storage mechanism:

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

Goal: General insights

Structure of storage $\Leftrightarrow$ computational properties

Storage mechanisms

Automata models that extend finite automata by some storage mechanism:

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

Goal: General insights

Structure of storage ⇔ computational properties

Framework

Abstract model with storage as parameter

Valence automata

Definition

A *monoid* is a set M with

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

Valence automata

Definition

A *monoid* is a set M with

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

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

Definition

A *monoid* is a set M with

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

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

Definition

A *monoid* is a set M with

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

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

Definition

A *monoid* is a set M with

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

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 decide emptiness?

Classical results can now be generalized:

Questions

- For which storage mechanisms can we decide emptiness?
- For which do we have a particular closure property?

Classical results can now be generalized:

Questions

- For which storage mechanisms can we decide emptiness?
- For which do we have a particular closure property?
- How is the complexity of decision problems affected?

Classical results can now be generalized:

Questions

- For which storage mechanisms can we decide emptiness?
- For which do we have a particular closure property?
- How is the complexity of decision problems affected?
- For which can we compute abstractions?

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

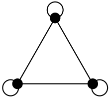
$$\begin{aligned} X_\Gamma &= \{a_v, \bar{a}_v \mid v \in V\} \\ R_\Gamma &= \{a_v\bar{a}_v = \varepsilon \mid v \in V\} \\ &\quad \cup \{xy = yx \mid x \in \{a_u, \bar{a}_u\}, y \in \{a_v, \bar{a}_v\}, \{u, v\} \in E\} \end{aligned}$$

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

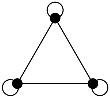
Intuition

- $\mathbb{B}$: 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

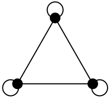


Examples



$\mathbb{Z}^3$

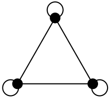
Examples



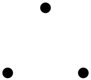
$\mathbb{Z}^3$

Blind counter

Examples

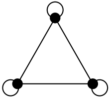


$\mathbb{Z}^3$

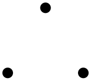


Blind counter

Examples



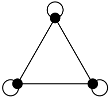
$\mathbb{Z}^3$



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

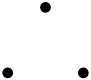
Blind counter

Examples



$\mathbb{Z}^3$

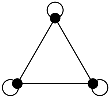
Blind counter



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

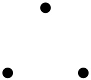
Pushdown

Examples



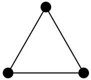
$\mathbb{Z}^3$

Blind counter

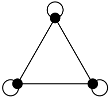


$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

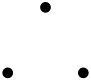


Examples



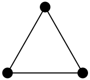
$\mathbb{Z}^3$

Blind counter



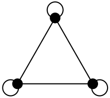
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



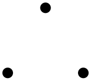
$\mathbb{B}^3$

Examples



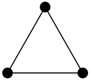
$\mathbb{Z}^3$

Blind counter



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

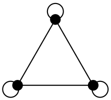
Pushdown



$\mathbb{B}^3$

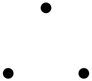
Partially blind counter

Examples



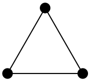
$\mathbb{Z}^3$

Blind counter



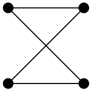
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

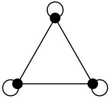


$\mathbb{B}^3$

Partially blind counter

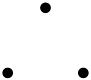


Examples



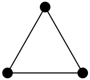
$\mathbb{Z}^3$

Blind counter



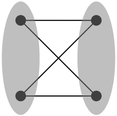
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

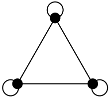


$\mathbb{B}^3$

Partially blind counter

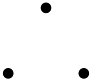


Examples



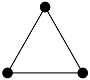
$\mathbb{Z}^3$

Blind counter



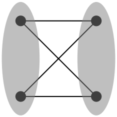
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



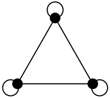
$\mathbb{B}^3$

Partially blind counter



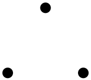
$(\mathbb{B} * \mathbb{B}) \times (\mathbb{B} * \mathbb{B})$

Examples



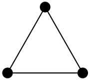
$\mathbb{Z}^3$

Blind counter



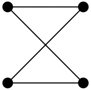
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



$\mathbb{B}^3$

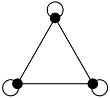
Partially blind counter



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

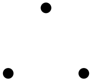
Infinite tape (TM)

Examples



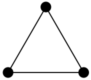
$\mathbb{Z}^3$

Blind counter



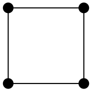
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



$\mathbb{B}^3$

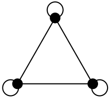
Partially blind counter



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

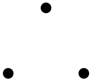
Infinite tape (TM)

Examples



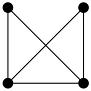
$\mathbb{Z}^3$

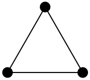
Blind counter



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

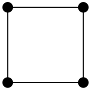
Pushdown





$\mathbb{B}^3$

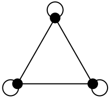
Partially blind counter



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

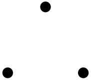
Infinite tape (TM)

Examples



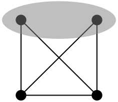
$\mathbb{Z}^3$

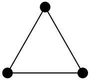
Blind counter



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

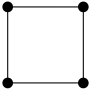
Pushdown





$\mathbb{B}^3$

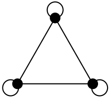
Partially blind counter



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

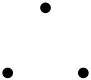
Infinite tape (TM)

Examples



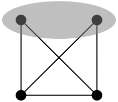
$\mathbb{Z}^3$

Blind counter

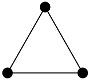


$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown

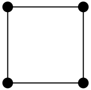


$(\mathbb{B} * \mathbb{B}) \times \mathbb{B} \times \mathbb{B}$



$\mathbb{B}^3$

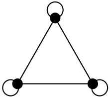
Partially blind counter



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

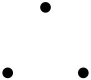
Infinite tape (TM)

Examples



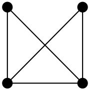
$\mathbb{Z}^3$

Blind counter



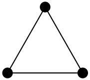
$\mathbb{B} * \mathbb{B} * \mathbb{B}$

Pushdown



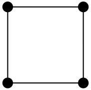
$(\mathbb{B} * \mathbb{B}) \times \mathbb{B} \times \mathbb{B}$

Pushdown + partially blind counters



$\mathbb{B}^3$

Partially blind counter



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

Infinite tape (TM)

The emptiness problem

The emptiness problem

Given a valence automaton over M, does it accept any word?

The emptiness problem

The emptiness problem

Given a valence automaton over M, does it accept any word?

Important problem

- Type of reachability problem
- Necessary for many other decision problems.

The emptiness problem

The emptiness problem

Given a valence automaton over M, does it accept any word?

Important problem

- Type of reachability problem
- Necessary for many other decision problems.

Question

For which storage mechanisms is the emptiness problem decidable?

The emptiness problem

The emptiness problem

Given a valence automaton over M, does it accept any word?

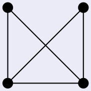
Important problem

- Type of reachability problem
- Necessary for many other decision problems.

Question

For which storage mechanisms is the emptiness problem decidable?

Obstacle



Pushdown + partially blind counters

The emptiness problem

The emptiness problem

Given a valence automaton over M, does it accept any word?

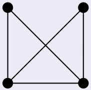
Important problem

- Type of reachability problem
- Necessary for many other decision problems.

Question

For which storage mechanisms is the emptiness problem decidable?

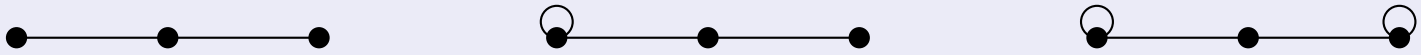
Obstacle



Pushdown + partially blind counters

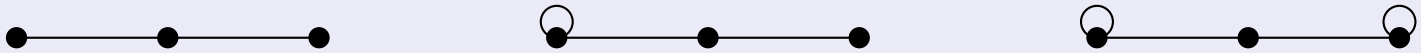
Decidability a long-standing open problem

Simplest graphs for pushdown + counters



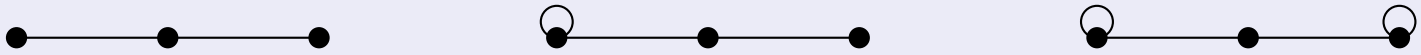
- One can show: These can simulate pushdown + one counter

Simplest graphs for pushdown + counters



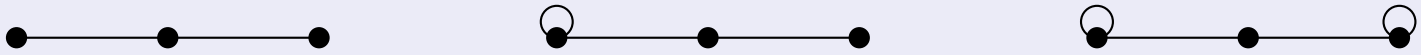
- One can show: These can simulate pushdown + one counter
- We call these *PPN-graphs* (for "pushdown Petri net").

Simplest graphs for pushdown + counters



- One can show: These can simulate pushdown + one counter
- We call these *PPN-graphs* (for "pushdown Petri net").
- Without them as induced subgraphs: *PPN-free*.

Simplest graphs for pushdown + counters



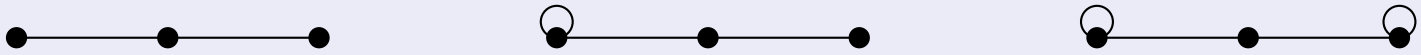
- One can show: These can simulate pushdown + one counter
- We call these *PPN-graphs* (for "pushdown Petri net").
- Without them as induced subgraphs: *PPN-free*.

Theorem

Let Γ *be PPN-free. Then the following are equivalent:*

- *Emptiness is decidable for valence automata over* $\mathbb{M}\Gamma$*.*
- Γ*, minus loops, is a transitive forest.*

Simplest graphs for pushdown + counters

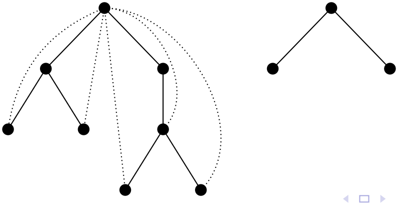


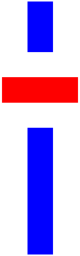
- One can show: These can simulate pushdown + one counter
- We call these *PPN-graphs* (for "pushdown Petri net").
- Without them as induced subgraphs: *PPN-free*.

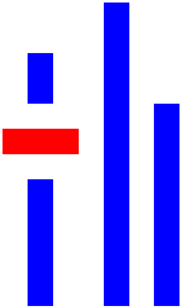
Theorem

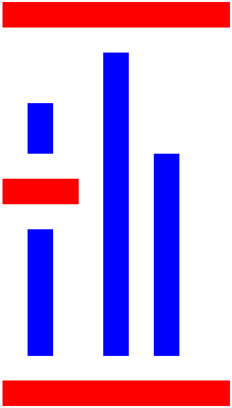
Let Γ be PPN-free. Then the following are equivalent:

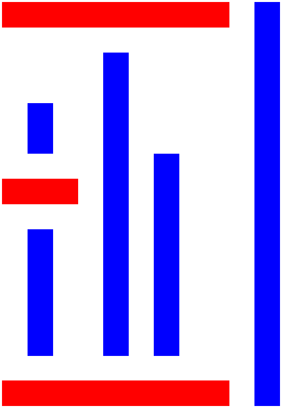
- *Emptiness is decidable for valence automata over* $\mathbb{M}\Gamma$.
- Γ, *minus loops, is a transitive forest.*

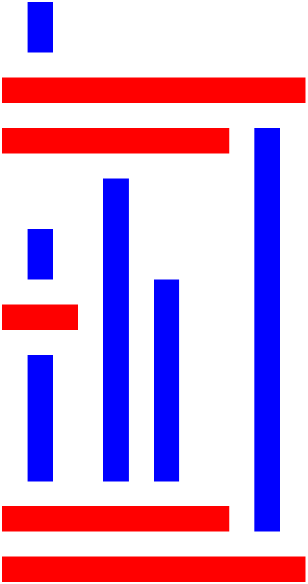


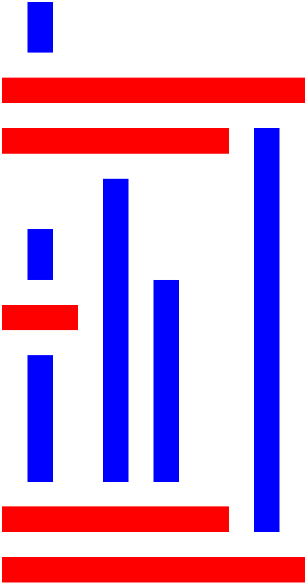








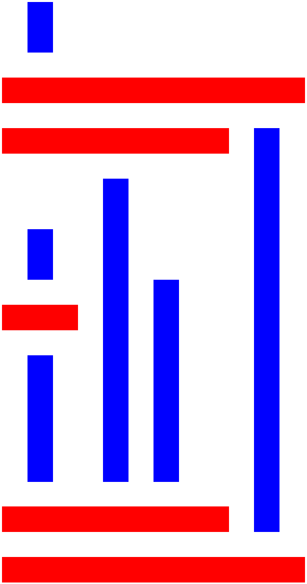




Intuition

Decidable mechanisms, $SC^{\pm}$:

- Start with partially blind counters
- Build stacks
- Add blind counters

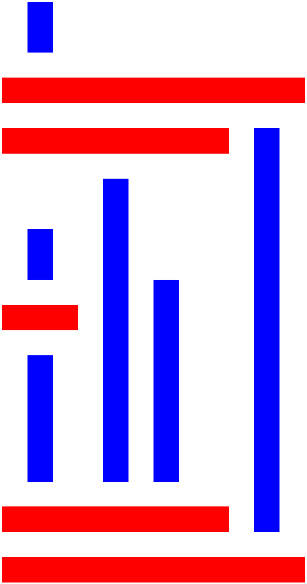


Intuition

Decidable mechanisms, $SC^{\pm}$:

- Start with partially blind counters
- Build stacks
- Add blind counters

$\Rightarrow$ Reduction to priority counter automata of Reinhardt



Intuition

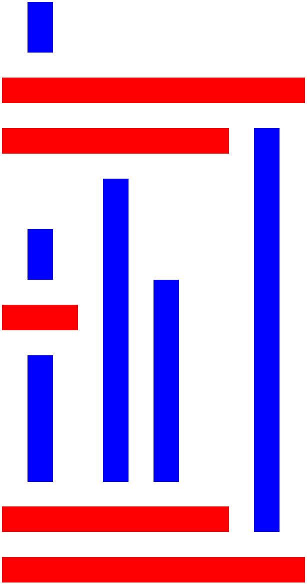
Decidable mechanisms, $SC^{\pm}$:

- Start with partially blind counters
- Build stacks
- Add blind counters

⇒ Reduction to priority counter automata of Reinhardt

Left open, SC^{+}:

- Start with partially blind counters
- Build stacks
- Add partially blind counters



Intuition

Decidable mechanisms, $SC^{\pm}$:

- Start with partially blind counters
- Build stacks
- Add blind counters

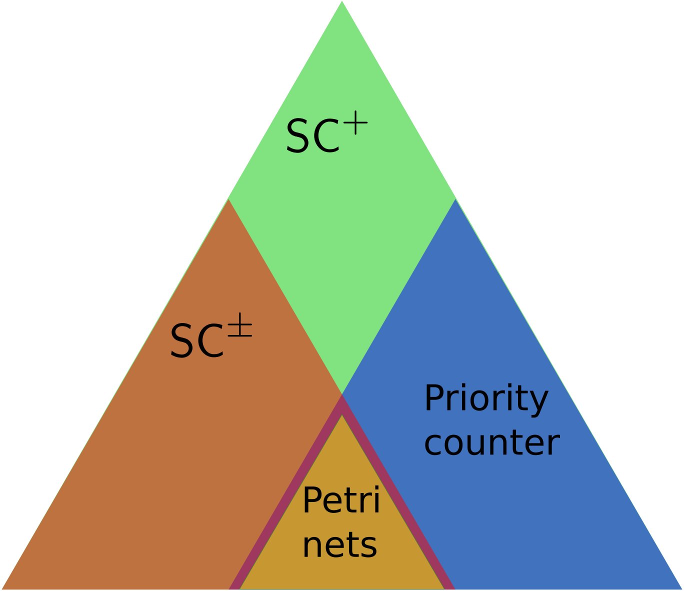
$\Rightarrow$ Reduction to priority counter automata of Reinhardt

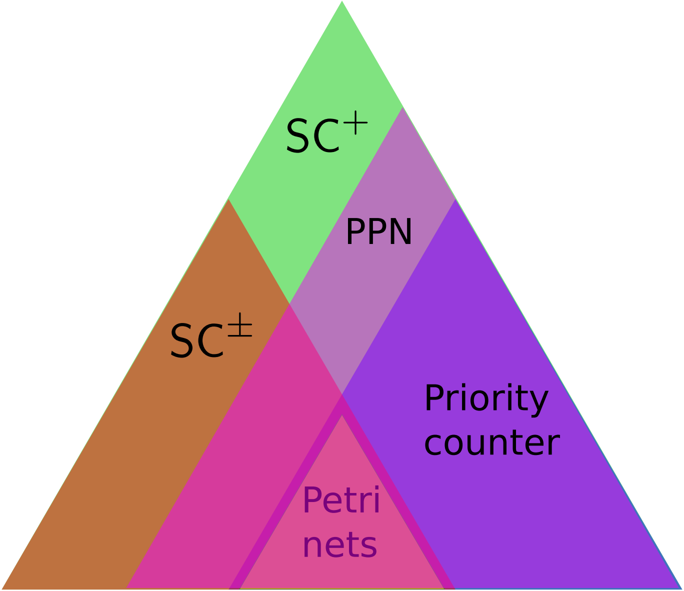
Left open, SC^{+}:

- Start with partially blind counters
- Build stacks
- Add partially blind counters

$\Rightarrow$ Generalize pushdown Petri nets and priority counter automata

$\Rightarrow$ New open problem

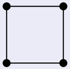

SC$^+$
SC$^\pm$
Priority counter
Petri nets


SC$^+$
PPN
SC$^\pm$
Priority counter
Petri nets

Poof: Undecidability

Theorem (Wolk 1965)

An undirected graph is a transitive forest iff it avoids as induced subgraphs:

C_4 : 

P_4 : 

$\Rightarrow$ Show Turing completeness for C_4 and P_4

Poof: Decidability

Decidability

Combinatorial argument shows: equivalent to $SC^{\pm}$.

Poof: Decidability

Decidability

Combinatorial argument shows: equivalent to $\mathsf{SC}^{\pm}$.

Definition of $\mathsf{SC}^{\pm}$

Smallest class with

- $\mathbb{B}^n \in \mathsf{SC}^{\pm}$
- if $M \in \mathsf{SC}^{\pm}$, then $\mathbb{B} * M, \quad \mathbb{Z} \times M \in \mathsf{SC}^{\pm}$

Poof: Decidability

Decidability

Combinatorial argument shows: equivalent to $\mathsf{SC}^{\pm}$.

Definition of $\mathsf{SC}^{\pm}$

Smallest class with

- $\mathbb{B}^n \in \mathsf{SC}^{\pm}$
- if $M \in \mathsf{SC}^{\pm}$, then $\mathbb{B} * M, \quad \mathbb{Z} \times M \in \mathsf{SC}^{\pm}$

Reduction

$\Psi(\mathsf{VA}(M)) \subseteq \mathsf{Prio}$ for every $M \in \mathsf{SC}^{\pm}$.

Priority counter machines

- Automaton with n counters

Priority counter machines

- Automaton with n counters
- counters stay $\geqslant 0$

Priority counter machines

- Automaton with n counters
- counters stay $\geqslant 0$
- instructions:
  - inc_i: increment counter i
  - dec_i: decrement counter i
  - zero_i: test all the counters $1, \ldots, i$ for zero

Priority counter machines

- Automaton with n counters
- counters stay $\geqslant 0$
- instructions:
  - inc_i: increment counter i
  - dec_i: decrement counter i
  - zero_i: test all the counters $1, \ldots, i$ for zero
- Language class: Prio

Priority counter machines

- Automaton with n counters
- counters stay $\geqslant 0$
- instructions:
  - inc_i: increment counter i
  - dec_i: decrement counter i
  - zero_i: test all the counters $1, \ldots, i$ for zero
- Language class: Prio

Theorem (Reinhardt)

Reachability is decidable for priority counter machines.

Definition of $\mathsf{SC}^{\pm}$

Smallest class with

- $\mathbb{B}^n \in \mathsf{SC}^{\pm}$
- if $M \in \mathsf{SC}^{\pm}$, then $\mathbb{B} * M, \quad \mathbb{Z} \times M \in \mathsf{SC}^{\pm}$

Observations

- $\mathsf{VA}(\mathbb{B}^n) \subseteq \mathsf{Prio}$, hence $\Psi(\mathsf{VA}(\mathbb{B}^n)) \subseteq \Psi(\mathsf{Prio})$.

Definition of $\mathsf{SC}^{\pm}$

Smallest class with

- $\mathbb{B}^n \in \mathsf{SC}^{\pm}$
- if $M \in \mathsf{SC}^{\pm}$, then $\mathbb{B} * M, \quad \mathbb{Z} \times M \in \mathsf{SC}^{\pm}$

Observations

- $\mathsf{VA}(\mathbb{B}^n) \subseteq \mathsf{Prio}$, hence $\Psi(\mathsf{VA}(\mathbb{B}^n)) \subseteq \Psi(\mathsf{Prio})$.
- If $\Psi(\mathsf{VA}(M)) \subseteq \mathsf{Prio}$, then $\Psi(\mathsf{VA}(M \times \mathbb{Z})) \subseteq \Psi(\mathsf{Prio})$.

Definition of $\mathsf{SC}^{\pm}$

Smallest class with

- $\mathbb{B}^n \in \mathsf{SC}^{\pm}$
- if $M \in \mathsf{SC}^{\pm}$, then $\mathbb{B} * M, \quad \mathbb{Z} \times M \in \mathsf{SC}^{\pm}$

Observations

- $\mathsf{VA}(\mathbb{B}^n) \subseteq \mathsf{Prio}$, hence $\Psi(\mathsf{VA}(\mathbb{B}^n)) \subseteq \Psi(\mathsf{Prio})$.
- If $\Psi(\mathsf{VA}(M)) \subseteq \mathsf{Prio}$, then $\Psi(\mathsf{VA}(M \times \mathbb{Z})) \subseteq \Psi(\mathsf{Prio})$.
- What about $\mathsf{VA}(\mathbb{B} * M)$?

Expressiveness

Algebraic extensions

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

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

Expressiveness

Algebraic extensions

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

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

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

Expressiveness

Algebraic extensions

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

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

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

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

Expressiveness

Algebraic extensions

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

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

Expressiveness

Algebraic extensions

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

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

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

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

Theorem (Z. 2015)

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

Expressiveness

Algebraic extensions

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

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

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

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

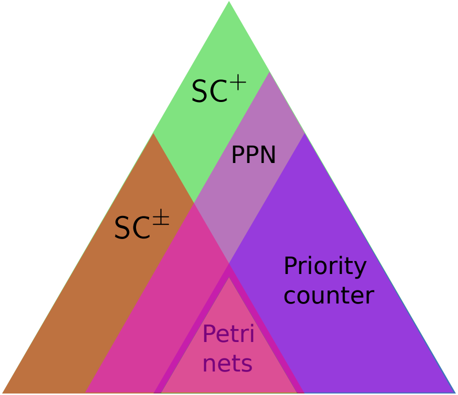
Theorem (Z. 2015)

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

Theorem (van Leeuwen 1974)

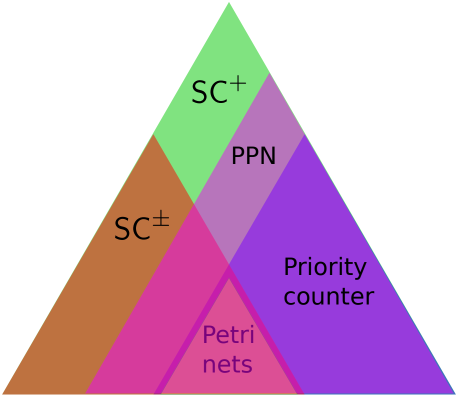
If $\mathcal{C}$ is closed under rational transductions and Kleene star, then $\Psi(\mathsf{Alg}(\mathcal{C})) \subseteq \Psi(\mathcal{C})$.

Conclusion



Contribution

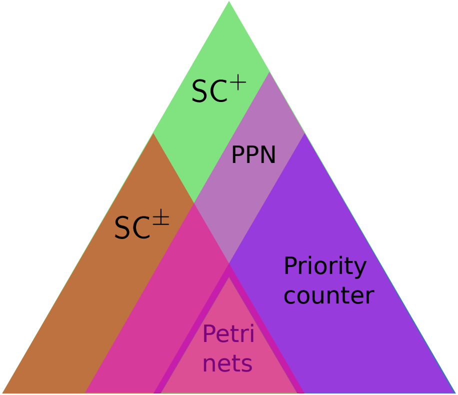
Conclusion



Contribution

- In absence of PPN: Characterization of decidable emptiness problem

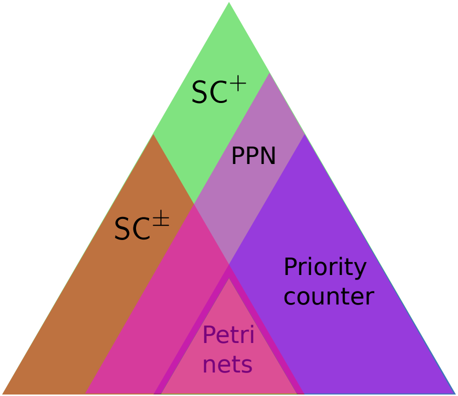
Conclusion



Contribution

- In absence of PPN: Characterization of decidable emptiness problem
- New decidable model ($SC^\pm$)

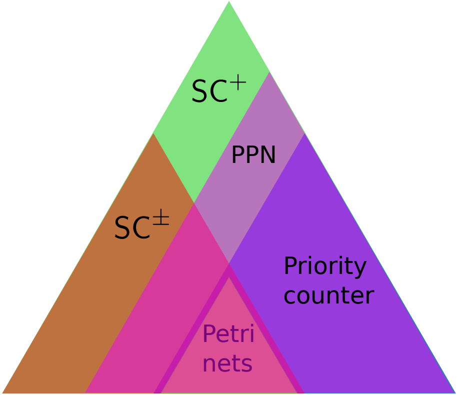
Conclusion



Contribution

- In absence of PPN: Characterization of decidable emptiness problem
- New decidable model (SC$^\pm$)
- New powerful model that might be decidable (SC$^+$)

Conclusion



Contribution

- In absence of PPN: Characterization of decidable emptiness problem
- New decidable model ($SC^\pm$)
- New powerful model that might be decidable (SC^+)