Low-Dimensional VASS: Where Our Tools Fail Us

Łukasz Orlikowski

University of Warsaw

04.06.2025

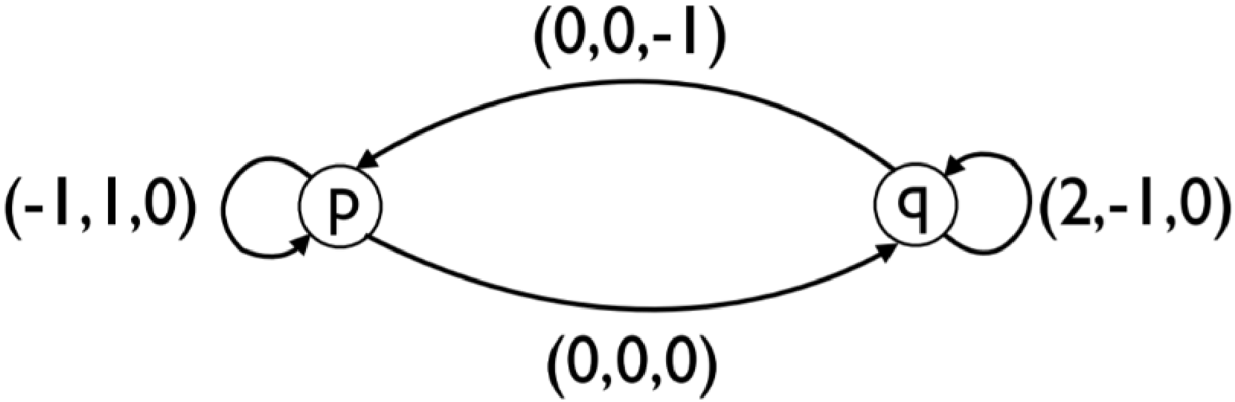
Outline

1. Introduction

Outline

Outline

Vector Additions Systems with States

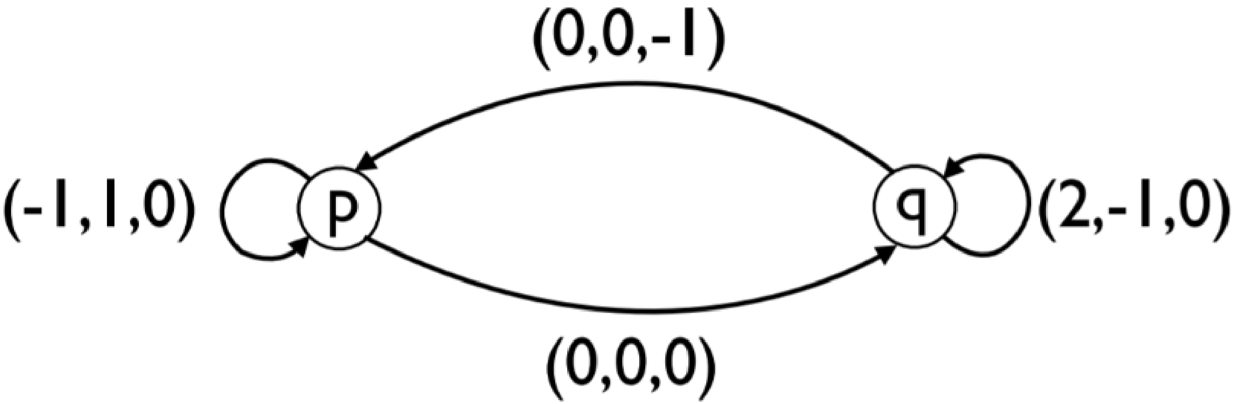


Reachability Problem

Given: A Vector Addition System with States (VASS) V, two configurations s and t.

Given: A Vector Addition System with States (VASS) V, two configurations s and t.

Question: Is there a run from s to t in V?



Short history

- Lipton ‘76: ExpSpace-hardness

Short history

- Lipton ‘76: ExpSpace-hardness
- Mayr ‘81: Decidability

Short history

- Lipton '76: ExpSpace-hardness
- Mayr '81: Decidability
- Kosaraju '82, Lambert '92: Simplifications

Short history

- Lipton ‘76: ExpSpace-hardness
- Mayr ‘81: Decidability
- Kosaraju ‘82, Lambert ‘92: Simplifications
- Leroux, Schmitz ‘19: Ackermann upper bound

Short history

- Lipton ‘76: ExpSpace-hardness
- Mayr ‘81: Decidability
- Kosaraju ‘82, Lambert ‘92: Simplifications
- Leroux, Schmitz ‘19: Ackermann upper bound
- Leroux and Czerwiński, O. ‘21: Ackermann-hardness

Low dimensional VASSs

	UNARY	BINARY
1	**NL-complete**	**NP-complete**

Low dimensional VASSs

	UNARY	BINARY
1	**NL-complete**	**NP-complete**
2	**NL-complete**	**PSPACE-complete**

Low dimensional VASSs

	UNARY	**BINARY**
1	**NL-complete**	**NP-complete**
2	**NL-complete**	**PSPACE-complete**
3	**NP-hard**	**PSPACE-hard, in 2-EXPSPACE**

Low dimensional VASSs

	UNARY	BINARY
1	NL-complete	NP-complete
2	NL-complete	PSPACE-complete
3	NP-hard	PSPACE-hard, in 2-EXPSPACE
4	NP-hard	PSPACE-hard

Low dimensional VASSs

	UNARY	BINARY
1	NL-complete	NP-complete
2	NL-complete	PSPACE-complete
3	NP-hard	PSPACE-hard, in 2-EXPSPACE
4	NP-hard	PSPACE-hard
5	PSPACE-hard	PSPACE-hard

Low dimensional VASSs

	UNARY	**BINARY**
1	**NL-complete**	**NP-complete**
2	**NL-complete**	**PSPACE-complete**
3	**NP-hard**	**PSPACE-hard, in 2-EXPSPACE**
4	**NP-hard**	**PSPACE-hard**
5	**PSPACE-hard**	**PSPACE-hard**
6	**PSPACE-hard**	**EXPSPACE-hard**

Low dimensional VASSs

	UNARY	**BINARY**
1	**NL-complete**	**NP-complete**
2	**NL-complete**	**PSPACE-complete**
3	**NP-hard**	**PSPACE-hard, in 2-EXPSPACE**
4	**NP-hard**	**PSPACE-hard**
5	**PSPACE-hard**	**PSPACE-hard**
6	**PSPACE-hard**	**EXPSPACE-hard**
8	**TOWER-hard**	**TOWER-hard**

Low dimensional VASSs

	UNARY	BINARY
1	NL-complete	NP-complete
2	NL-complete	PSPACE-complete
3	NP-hard	PSPACE-hard, in 2-EXPSPACE
4	NP-hard	PSPACE-hard
5	PSPACE-hard	PSPACE-hard
6	PSPACE-hard	EXPSPACE-hard
8	TOWER-hard	TOWER-hard

- Big gaps between lowerbounds and upperbounds

Low dimensional VASSs

	UNARY	BINARY
1	NL-complete	NP-complete
2	NL-complete	PSPACE-complete
3	NP-hard	PSPACE-hard, in 2-EXPSPACE
4	NP-hard	PSPACE-hard
5	PSPACE-hard	PSPACE-hard
6	PSPACE-hard	EXPSPACE-hard
8	TOWER-hard	TOWER-hard

- Big gaps between lowerbounds and upperbounds
- Hope for elementary complexity bounds

Low dimensional VASSs

	UNARY	BINARY
1	NL-complete	NP-complete
2	NL-complete	PSPACE-complete
3	NP-hard	PSPACE-hard, in 2-EXPSPACE
4	NP-hard	PSPACE-hard
5	PSPACE-hard	PSPACE-hard
6	PSPACE-hard	EXPSPACE-hard
8	TOWER-hard	TOWER-hard

- Big gaps between lowerbounds and upperbounds
- Hope for elementary complexity bounds
- Easier to develop techniques, which can be then generalised to the general case

Open problems

Open problem 1: What's the biggest dimension in which the reachability problem is in Elementary?

Open problems

Open problem 1: What's the biggest dimension in which the reachability problem is in Elementary?

Challenge: Finite reachability sets of not elementary size.

Open problems

Open problem 1: What's the biggest dimension in which the reachability problem is in Elementary?

Challenge: Finite reachability sets of not elementary size.

Big hopes: approximating reachability sets by semilinear sets (e.g. sandwiching technique).

Open problems

Open problem 1: What's the biggest dimension in which the reachability problem is in Elementary?

Challenge: Finite reachability sets of not elementary size.

Big hopes: approximating reachability sets by semilinear sets (e.g. sandwiching technique).

Open problem 2: Improve lowerbounds!

Open problems

Open problem 1: What's the biggest dimension in which the reachability problem is in Elementary?

Challenge: Finite reachability sets of not elementary size.

Big hopes: approximating reachability sets by semilinear sets (e.g. sandwiching technique).

Open problem 2: Improve lowerbounds!

Not clear how to proceed...

Bounded automaton

A B-bounded d-counter automaton A is an automaton with d-counters such that:

Bounded automaton

A B-bounded d-counter automaton A is an automaton with d-counters such that:

- Sum of all counters is bounded by B in each of the runs

Bounded automaton

A B-bounded d-counter automaton A is an automaton with d-counters such that:

- Sum of all counters is bounded by B in each of the runs
- A can decrease or increase counter by 1

Bounded automaton

A B-bounded d-counter automaton A is an automaton with d-counters such that:

- Sum of all counters is bounded by B in each of the runs
- A can decrease or increase counter by 1
- A can test counter to be zero

Bounded automaton

A B-bounded d-counter automaton A is an automaton with d-counters such that:

- Sum of all counters is bounded by B in each of the runs
- A can decrease or increase counter by 1
- A can test counter to be zero
- Each counter is nonnegative

The reachability problem for B-bounded d-counter automata

The reachability problem for B-bounded d-counter automata

Input: A number $n \in \mathbb{N}$ given in unary, a B-bounded d-counter automaton A with s states and states q_i, q_f

The reachability problem for B-bounded d-counter automata

Input: A number $n \in \mathbb{N}$ given in unary, a B-bounded d-counter automaton A with s states and states q_i, q_f

Question: Does A have an accepting run from $q_i(0,0)$ to $q_f(0,0)$?

The reachability problem for B-bounded d-counter automata

Input: A number $n \in \mathbb{N}$ given in unary, a B-bounded d-counter automaton A with s states and states q_i, q_f

Question: Does A have an accepting run from $q_i(0,0)$ to $q_f(0,0)$?

- PSPACE-hard when $B = 2^n$ and $d = 2$

Complexity

The reachability problem for B-bounded d-counter automata

Input: A number $n \in \mathbb{N}$ given in unary, a B-bounded d-counter automaton A with s states and states q_i, q_f

Question: Does A have an accepting run from $q_i(0,0)$ to $q_f(0,0)$?

- PSPACE-hard when $B = 2^n$ and $d = 2$
- EXSPACE-hard when $B = 2^{2^n}$ and $d = 3$

The reachability problem for B-bounded d-counter automata

Input: A number $n \in \mathbb{N}$ given in unary, a B-bounded d-counter automaton A with s states and states q_i, q_f

Question: Does A have an accepting run from $q_i(0,0)$ to $q_f(0,0)$?

- PSPACE-hard when $B = 2^n$ and $d = 2$
- EXSPACE-hard when $B = 2^{2^n}$ and $d = 3$
- Not known when $B = 2^{2^n}$ and $d = 2$

Techniques

Technique	Pros ✓	Cons ✗
Controlling counter	Simple, One counter	Polynomially many zero-tests

Techniques

Technique	Pros ✓	Cons ✗
Controlling counter	Simple, One counter	Polynomially many zero-tests
Multiplication triple (B, C, BC)	Relatively easy to produce	Three counters

Techniques

Technique	Pros ✓	Cons ✗
Controlling counter	Simple, One counter	Polynomially many zero-tests
Multiplication triple (B, C, BC)	Relatively easy to produce	Three counters
Quadratic pair (B, B^2)	Two counters	Harder to produce

Controlling counter

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \dots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n$$

- x_i: the value of the counter in state c_i

Controlling counter

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \ldots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n$$

- x_i: the value of the counter in state c_i
- We know, that $x_i \geq 0$.

Controlling counter

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \dots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n$$

- x_i: the value of the counter in state c_i
- We know, that $x_i \geq 0$.
- Hence, it is enough to check $x_1 + x_2 + \dots x_n = 0$

Controlling counter

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \ldots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n$$

- x_i: the value of the counter in state c_i
- We know, that $x_i \geq 0$.
- Hence, it is enough to check $x_1 + x_2 + \ldots x_n = 0$
- x_i': the effect of the run ρ_i on the counter x

Controlling counter

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \ldots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n$$

- x_i: the value of the counter in state c_i
- We know, that $x_i \geq 0$.
- Hence, it is enough to check $x_1 + x_2 + \ldots x_n = 0$
- x'_i: the effect of the run ρ_i on the counter x
- Observe, that $x_i = x'_1 + x'_2 + \ldots + x'_i$

Controlling counter

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \ldots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n$$

- x_i: the value of the counter in state c_i
- We know, that $x_i \geq 0$.
- Hence, it is enough to check $x_1 + x_2 + \ldots x_n = 0$
- x'_i: the effect of the run ρ_i on the counter x
- Observe, that $x_i = x'_1 + x'_2 + \ldots + x'_i$

To verify that $x_i = 0$ for all i, we introduce a new counter:

$$c = x_1 + x_2 + \cdots + x_n = nx'_1 + \cdots + x'_n.$$

Limitations

- 2 or 3 counters coming from counter automata

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test
- at least one counter to produce pair or triple used for zero-testing

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test
- at least one counter to produce pair or triple used for zero-testing
- total: at least 5 counters

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test
- at least one counter to produce pair or triple used for zero-testing
- total: at least 5 counters
- **Question 1:** Can we do better? (for instance share counters)

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test
- at least one counter to produce pair or triple used for zero-testing
- total: at least 5 counters
- **Question 1:** Can we do better? (for instance share counters)
- **Question 2:** Can we devise a technique using fewer counters for zero-testing?

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test
- at least one counter to produce pair or triple used for zero-testing
- total: at least 5 counters
- **Question 1:** Can we do better? (for instance share counters)
- **Question 2:** Can we devise a technique using fewer counters for zero-testing?
- **Question 3:** Can we find a better problem to reduce from?

Limitations

- 2 or 3 counters coming from counter automata
- 2 or 3 counters to simulate zero test
- at least one counter to produce pair or triple used for zero-testing
- total: at least 5 counters
- **Question 1:** Can we do better? (for instance share counters)
- **Question 2:** Can we devise a technique using fewer counters for zero-testing?
- **Question 3:** Can we find a better problem to reduce from?

Thank You!