Undecidability of Two-dimensional Robot Games

Reino Niskanen\textsuperscript{1}  Igor Potapov\textsuperscript{1},  Julien Reichert\textsuperscript{2}

\textsuperscript{1}Department of Computer Science
University of Liverpool, UK

\textsuperscript{2}LSV, ENS Cachan, France

41st International Symposium on Mathematical Foundations of Computer Science
Introduction
Graph reachability

Drive from the factory back home.
Graph reachability

Drive from the factory back home.
Graph reachability

Drive from the factory back home.
Graph reachability

Drive from the factory back home.
Graph reachability

Drive from the factory back home.
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly $k$. 

```
Drive home from the factory.
Weights tell how much profit you make on the route.
For example, the goal is to have profit of exactly $k$.
```
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly $k$. 

```
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly $k$.
```
Weighted graph reachability

Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly $k$. 
Weighted graph reachability

Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly $k$. 

Niskanen, Potapov, and Reichert

Two-dimensional Robot Games

MFCS 2016
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
A counter reachability game

Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.
Definitions
Counter reachability games

- Played on a labeled directed graph $G = (V, E)$ with edges labeled by $x \in \mathbb{Z}^n$.
- Two players: Eve (◯), Adam (□).
- A configuration $[v, z] \in V \times \mathbb{Z}^n$.
- A successor configuration is $[v', z + z']$, where $[v, z', v'] \in E$ and the owner of $v$ chose it.
- The initial and target configurations.
- A play is a finite or an infinite sequence of configurations.
- Eve wins if the target configuration is reachable in a play starting from the initial configuration. Otherwise Adam wins.
Counter reachability games

A winning strategy

Eve has a winning strategy if the target configuration is reachable for every choice of Adam.

The decision problem

Given a graph $G = (V, E)$, initial and target configurations, $[v_0, z_0]$ and $[v_f, (0, \ldots, 0)]$. Does there exist a winning strategy for Eve?
Robot games

Known for counter reachability games

<table>
<thead>
<tr>
<th>One-dimensional</th>
<th>EXPSPACE-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-dimensional</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>

What if we have a simpler graph?

- \textbf{EXPTIME}-complete in dimension one [Arul, Reichert, QAPL 2013].
- Undecidable in dimension three [Reichert, PhD thesis 2015].
- Remained open in dimension two.
Another way to look at robot games

- Played on integer lattice $\mathbb{Z}^n$.
- Adam and Eve move a token on the lattice.
- Eve’s goal is to reach $(0, \ldots, 0)$. Adam’s goal is to avoid it.

Adam’s moves: $A = \{(1, 2), (2, 0)\}$

Eve’s moves: $E = \{(2, 2), (1, 4)\}$
Another way to look at robot games

- Played on integer lattice $\mathbb{Z}^n$.
- Adam and Eve move a token on the lattice.
- Eve’s goal is to reach $(0, \ldots, 0)$. Adam’s goal is to avoid it.

Adam’s moves: $A = \{(1, 2), (2, 0)\}$

Eve’s moves: $E = \{(2, 2), (1, 4)\}$
Another way to look at robot games

- Played on integer lattice \( \mathbb{Z}^n \).
- Adam and Eve move a token on the lattice.
- Eve’s goal is to reach \((0, \ldots, 0)\). Adam’s goal is to avoid it.

Adam’s moves: \( A = \{(1, 2), (2, 0)\} \)

Eve’s moves: \( E = \{(2, 2), (1, 4)\} \)
Another way to look at robot games

- Played on integer lattice $\mathbb{Z}^n$.
- Adam and Eve move a token on the lattice.
- Eve’s goal is to reach $(0, \ldots, 0)$. Adam’s goal is to avoid it.

Adam’s moves: $A = \{ (1, 2), (2, 0) \}$

Eve’s moves: $E = \{ (2, 2), (1, 4) \}$
Another way to look at robot games

- Played on integer lattice $\mathbb{Z}^n$.
- Adam and Eve move a token on the lattice.
- Eve’s goal is to reach $(0, \ldots, 0)$. Adam’s goal is to avoid it.

Adam’s moves: $A = \{(1, 2), (2, 0)\}$

Eve’s moves: $E = \{(2, 2), (1, 4)\}$
Another way to look at robot games

- Played on integer lattice $\mathbb{Z}^n$.
- Adam and Eve move a token on the lattice.
- Eve’s goal is to reach $(0, \ldots, 0)$. Adam’s goal is to avoid it.

Adam’s moves: $A = \{(1, 2), (2, 0)\}$

Eve’s moves: $E = \{(2, 2), (1, 4)\}$

**Theorem**

Given moves of Adam and Eve, $A, E \subseteq \mathbb{Z}^2$, an initial vector $x \in \mathbb{Z}^2$. It is undecidable whether Eve has a winning strategy.
To prove the main result, we consider an extension.

Robot games but players have internal states as well.

**EXPSPACE**-complete in dimension one [to be presented at RP’16].
To prove the main result, we consider an extension.
Robot games but players have internal states as well.
\textbf{EXPSPACE}-complete in dimension one [to be presented at RP'16].

\textbf{Theorem}

\textit{It is undecidable which player wins in a two-dimensional robot game with states.}
Robot games with states
The basis of undecidability proofs

Deterministic two-counter Minsky machine (2CM):

- Two counters, $c_1$ and $c_2$.
- $m$ instructions: $1 : \text{INS}_1, \ldots, m : \text{INS}_m$, where $\text{INS}_i$ is
  - $i : c_1++; \text{ goto } k$, or
  - $i : c_2++; \text{ goto } k$, or
  - $i : \text{ if } c_1=0 \text{ goto } k \text{ else } c_1--; \text{ goto } j$, or
  - $i : \text{ if } c_2=0 \text{ goto } k \text{ else } c_2--; \text{ goto } j$, or
  - $i : \text{ halt}$.

- The halting problem is undecidable [Minsky, 1967].
2CM example

A Minsky machine that halts on an input \((x, y)\) iff \(x = 2^k\) (regardless of \(y\)).

1: \(\text{if } c_2 = 0 \text{ goto 2 else } c_2--; \text{ goto 1}\)
2: \(\text{if } c_1 = 0 \text{ goto 5 else } c_1--; \text{ goto 3}\)
3: \(\text{if } c_1 = 0 \text{ goto 7 else } c_1--; \text{ goto 4}\)
4: \(c_2++; \text{ goto 2}\)
5: \(\text{if } c_2 = 0 \text{ goto 2 else } c_2--; \text{ goto 6}\)
6: \(c_1++; \text{ goto 5}\)
7: \(\text{if } c_2 = 0 \text{ goto 8 else } c_2--; \text{ goto 9}\)
8: \(\text{halt}\)
9: \(c_1++; \text{ goto 9}\)
Eve simulates transitions of the machine.
Adam verifies that zero-checks are performed correctly.
Eve simulates transitions of the machine.
Adam verifies that zero-checks are performed correctly.

To simulate zero-checks, we increase the state space and store information on positivity of counters in the states.

Additionally, we multiply all the values in the first dimension by four to create extra space.
Two-counter machines to robot games with states

1: if \( c_2 = 0 \) goto 2 else \( c_2--; \) goto 1
2: if \( c_1 = 0 \) goto 5 else \( c_1--; \) goto 3
3: if \( c_1 = 0 \) goto 7 else \( c_1--; \) goto 4
4: \( c_2++; \) goto 2
5: if \( c_2 = 0 \) goto 2 else \( c_2--; \) goto 6
6: \( c_1++; \) goto 5
7: if \( c_2 = 0 \) goto 8 else \( c_2--; \) goto 9
8: halt
9: \( c_1++; \) goto 9
Two-counter machines to robot games with states

1: if $c_2=0$ goto 2 else $c_2--;$ goto 1
2: if $c_1=0$ goto 5 else $c_1--;$ goto 3
3: if $c_1=0$ goto 7 else $c_1--;$ goto 4
4: $c_2++;$ goto 2
5: if $c_2=0$ goto 2 else $c_2--;$ goto 6
6: $c_1++;$ goto 5
7: if $c_2=0$ goto 8 else $c_2--;$ goto 9
8: halt
9: $c_1++;$ goto 9
Two-counter machines to robot games with states

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

\[[3_{+0}, (8, 0)]\]

state

counters
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

[3_{+0}, (8, 0)]

state
counters

Niskanen, Potapov, and Reichert
Two-dimensional Robot Games
MFCS 2016
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

$[3_{+0}, (8, 0)] \rightarrow [4_{+0}, (4, 0)]$

state match

counters
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

[3_{+0}, (8, 0)]

Eve’s moves:
SIMULATION of 2CM (correct/incorrect)
EMPTYING MOVE

Adam’s moves:
0-MOVE
POSITIVITY CHECK
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

[3_{+0}, (8, 0)]

state

counters
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$

Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

$[3_{+0}, (8, 0)] \rightarrow [4_{00}, (4, 0)]$

state don’t match
counters
Two-counter machines to robot games with states

\[ [3_{+0}, (8, 0)] \rightarrow [4_{00}, (4, 0)] \rightarrow [4_{00}, (5, 0)] \]

state \[ \text{don't match} \]

 counters

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

Eve wins if 2CM reaches \( Q \times (0, 0) \)
Adam wins if 2CM does not reach \( Q \times (0, 0) \)
Two-counter machines to robot games with states

\[
\begin{align*}
[3_{+0}, (8, 0)] & \rightarrow [4_{00}, (4, 0)] \rightarrow [4_{00}, (5, 0)]
\end{align*}
\]

state \text{ doesn’t match}
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

$[3_{+0}, (8, 0)] \rightarrow [4_{00}, (4, 0)] \rightarrow [4_{00}, (5, 0)] \rightarrow [\top_{00}, (4, 0)]$

state

counters
don’t match

Eve cannot reach (0,0)
Two-counter machines to robot games with states

Adam wins if 2CM does not reach $Q \times (0, 0)$
Eve wins if 2CM reaches $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

$[3_{+0}, (8, 0)]$
Two-counter machines to robot games with states

Adam wins if 2CM reaches $Q \times (0, 0)$

Eve wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

$[3_{+0}, (8, 0)]$

state
counters
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

$[3_{+0}, (8, 0)] \rightarrow [4_{+0}, (4, 0)]$

state match counters
Two-counter machines to robot games with states

Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve wins if 2CM reaches $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

\[
[3_{+0}, (8, 0)] \rightarrow [4_{+0}, (4, 0)] \rightarrow [4_{+0}, (5, 0)]
\]

state

match

counters

Niskanen, Potapov, and Reichert

Two-dimensional Robot Games

MFCS 2016
Two-counter machines to robot games with states

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

[3_{+0}, (8, 0)] \rightarrow [4_{+0}, (4, 0)] \rightarrow [4_{+0}, (5, 0)]
Two-counter machines to robot games with states

Adam wins if 2CM reaches $Q \times (0, 0)$
Eve wins if 2CM does not reach $Q \times (0, 0)$

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam wins

Eve wins

positivity check

emptying move

state match still match

counters

match

still match

$[3_{+0}, (8, 0)] \rightarrow [4_{+0}, (4, 0)] \rightarrow [4_{+0}, (5, 0)] \rightarrow [\top_{+0}, (4, 0)]$
Two-counter machines to robot games with states

Niskanen, Potapov, and Reichert

Two-dimensional Robot Games

MFCS 2016
Two-counter machines to robot games with states

Adam wins if 2CM reaches $Q \times (0, 0)$

Eve wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

[3_{+0}, (8, 0)]

state counters
Introduction

Definitions

Robot games with states

Robot games

Conclusion

Two-counter machines to robot games with states

Adam wins

Eve wins

positivity check

emptying move

simulation (correct)

simulation (incorrect)

[3_{+0}, (8, 0)] \rightarrow [T_{+0}, (7, 0)]

state

counters

not 0 (mod 4)
Two-counter machines to robot games with states

Adam wins if 2CM does not reach $Q \times (0, 0)$
Eve wins if 2CM reaches $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

\[
[3_{+0}, (8, 0)] \rightarrow [\top_{+0}, (7, 0)] \rightarrow [\top_{+0}, (7, 0)]
\]

state

not 0 (mod 4)

counters
Two-counter machines to robot games with states

Adam wins if 2CM reaches $Q \times (0, 0)$
Eve wins if 2CM does not reach $Q \times (0, 0)$

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Eve wins if 2CM reaches $Q \times (0, 0)$
Adam wins if 2CM does not reach $Q \times (0, 0)$

Eve’s moves:
- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam’s moves:
- 0-MOVE
- POSITIVITY CHECK

\[ [3_{+0}, (8, 0)] \rightarrow [T_{+0}, (7, 0)] \rightarrow [T_{+0}, (7, 0)] \]

state

counters

not 0 (mod 4)
Robot games
The main challenge is to encode the state structure into integers.

Eve and Adam simulate moves in robot game with states.

Adam verifies that Eve moves according to the state structure.
Robot games with states to robot games

- The main challenge is to encode the state structure into integers.
- Eve and Adam simulate moves in robot game with states.
- Adam verifies that Eve moves according to the state structure.
Robot games with states to robot games

- The main challenge is to encode the state structure into integers.
- Eve and Adam simulate moves in robot game with states.
- Adam verifies that Eve moves according to the state structure.
The main challenge is to encode the state structure into integers.

Eve and Adam simulate moves in robot game with states.

Adam verifies that Eve moves according to the state structure.
Removing the states from robot games with states

A move in 2RGS: \( (x, y) \) from \( i \) to \( j \)

The corresponding move in 2RG: \( (x, yN - 2^i + 2^j) \), where \( N \in \mathbb{N} \).
Removing the states from robot games with states

A move in 2RGS:

\[ i \xrightarrow{(x,y)} j \]

The corresponding move in 2RG:

\[ (x, yN - 2^i + 2^j), \text{ where } N \in \mathbb{N}. \]

- Too simple: Several wrong moves can result in a right one.
Removing the states from robot games with states

A move in 2RGS: $(x, y)$

The corresponding move in 2RG:

$$(x, yN - 2^i + 2^j), \text{ where } N \in \mathbb{N}.$$
Robot games with states to robot games

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
\[(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)\]

emptying states

states of 2RGS
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
(1, 0 · 4 · 8^{10} + 0 · 8^{9} + 0 · 8^{8} + 0 · 8^{7} + 0 · 8^{6} + 0 · 8^{5} + 0 · 8^{4} + 0 · 8^{3} + 0 · 8^{2} + 1 · 8^{1} − 1 · 8^{0})
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
\[ (1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0 \) \]

emptying states
\[ (1, -8^1 + 8^2) \]

states of 2RGS
\[ (2, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0) \]
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
\[(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)\]

Emptying states
\[(1, 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)\]
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
(1, 0 \cdot 4^10 + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
(1, 0 · 4 · 8^{10} + 0 · 8^9 + 0 · 8^8 + 0 · 8^7 + 0 · 8^6 + 0 · 8^5 + 0 · 8^4 + 0 · 8^3 + 0 · 8^2 + 1 · 8^1 − 1 · 8^0)

emptying states
↓ (0, 1 · 4 · 8^{10} − 8^2 + 8^3)

states of 2RGS
(1, 1 · 4 · 8^{10} + 0 · 8^9 + 0 · 8^8 + 0 · 8^7 + 0 · 8^6 + 0 · 8^5 + 0 · 8^4 + 1 · 8^3 − 1 · 8^2 + 1 · 8^1 − 1 · 8^0)

Niskanen, Potapov, and Reichert
Two-dimensional Robot Games
MFCS 2016  21 / 25
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters

\[
\begin{align*}
(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
(1, 1 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 1 \cdot 8^3 - 1 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
(1, 1 \cdot 3 \cdot 8^{10} - 5 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 1 \cdot 8^3 - 1 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)
\end{align*}
\]
Robot games with states to robot games

1

(1, 0)

2

(0, 1)

3

Adam wins

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters

emptying states

states of 2RGS

$T_{00}$

$\begin{align*}
(1, 0 & \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
\downarrow & (0, 1 \cdot 4 \cdot 8^{10} - 8^2 + 8^3) \\
(1, 1 & \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 1 \cdot 8^3 - 1 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
\downarrow & (0, -8^{10} - 5 \cdot 8^9) \\
(1, 1 & \cdot 3 \cdot 8^{10} - 5 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 1 \cdot 8^3 - 1 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)
\end{align*}$

Simon Niskanen, Alexander Potapov, and Mathias Reichert

Two-dimensional Robot Games

MFCS 2016
Robot games with states to robot games

Adam wins if
Eve wins in 2RGS

Eve wins if
Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters

\[
(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)
\]
Robot games with states to robot games

1. Introduction
2. Definitions
3. Robot games with states
4. Robot games
5. Conclusion

### Robot games with states to robot games

#### Graphical Representation

- **Eve** wins if Eve wins in 2RGS
- **Adam** wins if Adam wins in 2RGS

#### States and Moves

- **Eve**'s moves:
  - SIMULATION of 2RGS (correct/incorrect)
  - STATE-DEFENCE MOVE
- **Adam**'s moves:
  - REGULAR MOVE
  - STATE-CHECK

#### Mathematical Expressions

- **2RGS counters**
  - \((1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)\)
- **Emptying states**
  - \((0, -8^{10} - 5 \cdot 8^9)\)
- **States of 2RGS**
  - \(T_{00}\)
Robot games with states to robot games

1. **Simulation (incorrect)**
   - Eve wins if Eve wins in 2RGS
   - Adam wins if Adam wins in 2RGS

2. **Simulation (correct)**
   - Eve wins if Eve wins in 2RGS
   - Adam wins if Adam wins in 2RGS

3. **State-check**
   - Regular move
   - State-defence move

4. **Eve’s moves:**
   - Simulation of 2RGS (correct/incorrect)
   - State-defence move

5. **Adam’s moves:**
   - Regular move
   - State-check

6. **2RGS counters**
   \[
   (1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)
   \]

7. **Emptying states**
   \[
   (1, -1 \cdot 8^{10} - 5 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)
   \]

8. **States of 2RGS**
   \[
   (1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 1 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0)
   \]
Robot games with states to robot games

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
\[(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)\]
\[\downarrow (0, 8^{10} + 5 \cdot 8^9 + 8^5 - 1 \cdot 8^1)\]

Emptying states
\[(1, -1 \cdot 8^{10} - 5 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)\]
\[\downarrow (0, 8^{10} + 5 \cdot 8^9 + 8^5 - 1 \cdot 8^1)\]

States of 2RGS
\[(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 1 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0)\]
Robot games with states to robot games

Adam wins

Eve wins

Eve wins if Eve wins in 2RGS
Adam wins if Adam wins in 2RGS

state-check
state-defence move

regular move

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters

(1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)

emptying states

states of 2RGS

Niskanen, Potapov, and Reichert
Two-dimensional Robot Games
MFCS 2016
Robot games with states to robot games

1. Adam wins
2. Eve wins
3. Adam wins

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters

\[ (1, 0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0 ) \]

emptying states

\[ (0, 8^{10} + 5 \cdot 8^9 + 8^5 - 1 \cdot 8^1 ) \]

states of 2RGS

\[ (1, 1 \cdot 8^{10} + 5 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 1 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0 ) \]
Robot games with states to robot games

Adam wins if Adam wins in 2RGS
Eve wins if Eve wins in 2RGS

Eve's moves:
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:
- REGULAR MOVE
- STATE-CHECK

2RGS counters
- emptying states
- states of 2RGS

\[
(1, 4 \cdot 8^{10} + 0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
(1, 1 \cdot 8^{10} + 5 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 1 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0) \\
(0, 8^{10} + 5 \cdot 8^9 + 8^5 - 1 \cdot 8^1) \\
\]
Conclusion
Theorem

Given moves of Adam and Eve, $A, E \subseteq \mathbb{Z}^2$, an initial vector $x \in \mathbb{Z}^2$. It is undecidable whether Eve has a winning strategy.

<table>
<thead>
<tr>
<th>Game</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter reachability games</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>robot games with states</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>robot games</td>
<td>EXPTIME-complete</td>
</tr>
</tbody>
</table>
Given moves of Adam and Eve, $A, E \subseteq \mathbb{Z}^2$, an initial vector $x \in \mathbb{Z}^2$. It is undecidable whether Eve has a winning strategy.

<table>
<thead>
<tr>
<th>Game</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$1$</td>
</tr>
<tr>
<td>counter reachability games</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>robot games with states</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>robot games</td>
<td>EXPTIME-complete</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$2$</th>
<th>$\geq 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter reachability games</td>
<td>U</td>
<td>–</td>
</tr>
<tr>
<td>robot games with states</td>
<td>U</td>
<td>–</td>
</tr>
<tr>
<td>robot games</td>
<td>U</td>
<td>–</td>
</tr>
</tbody>
</table>
Future work

- Better bounds on number of moves for each player.
- Embedding two-counter machines into different games.
- Decidability of stateless VASS games.
Thank you for your attention!