Undecidability of Two-dimensional Robot Games

Reino Niskanen[1] Igor Potapov[1], Julien Reichert[2]

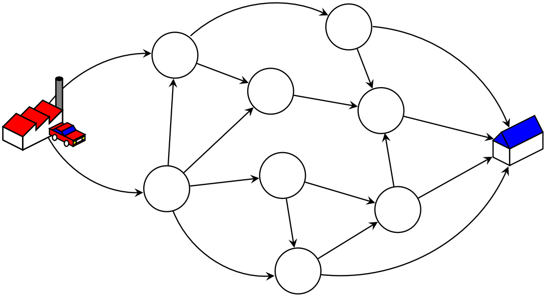
[1]Department of Computer Science
University of Liverpool, UK

[2]LSV, ENS Cachan, France

41st International Symposium on Mathematical Foundations of Computer Science

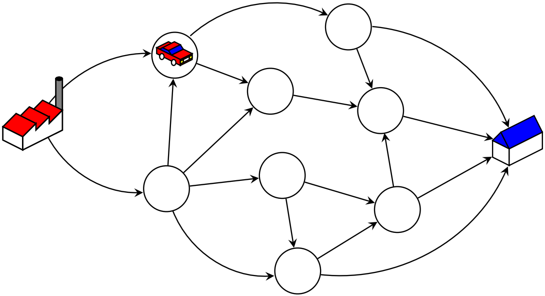
Introduction

Graph reachability



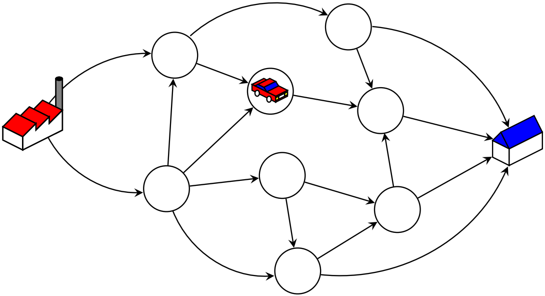
Drive from the factory back home.

Graph reachability



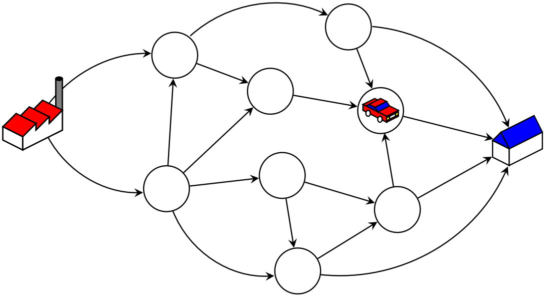
Drive from the factory back home.

Graph reachability



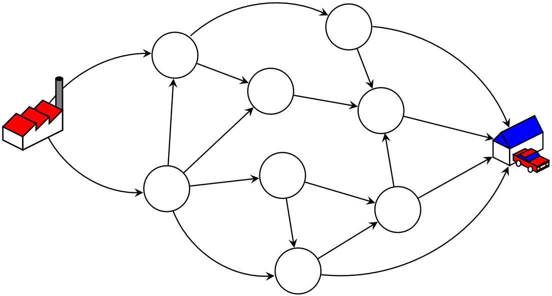
Drive from the factory back home.

Graph reachability



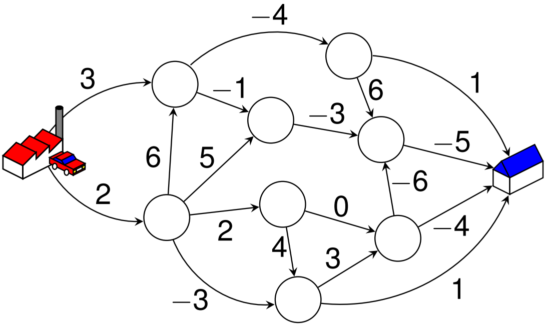
Drive from the factory back home.

Graph reachability



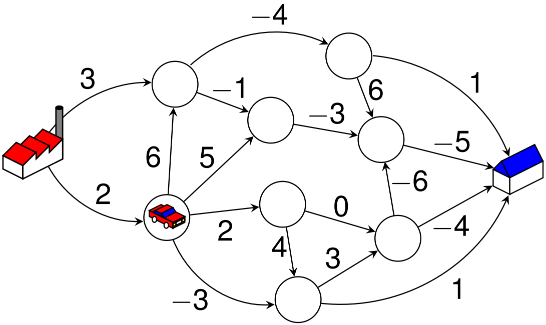
Drive from the factory back home.

Weighted graph reachability



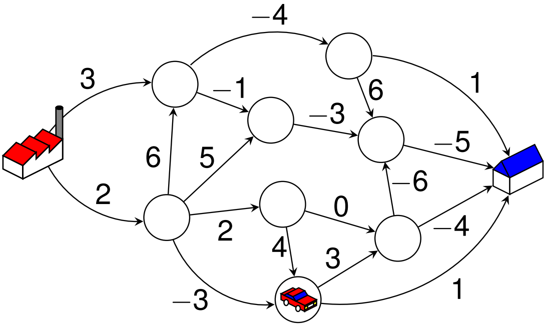
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly k.

Weighted graph reachability



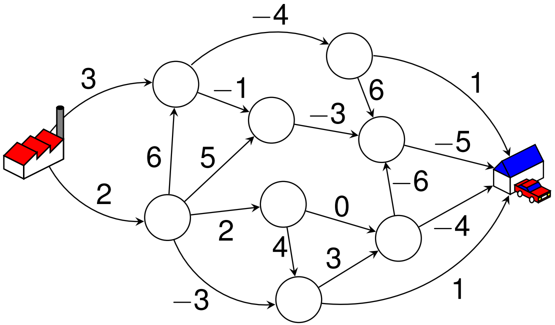
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly k.

Weighted graph reachability



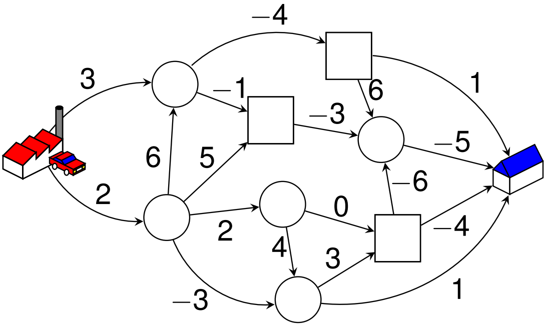
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly k.

Weighted graph reachability



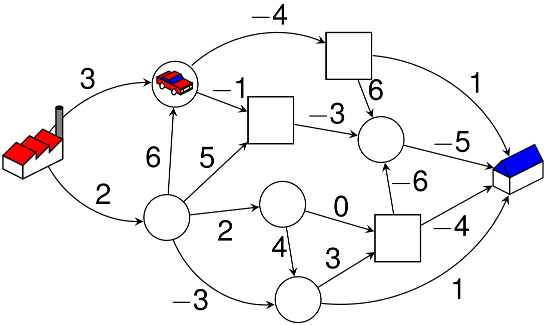
Drive home from the factory. Weights tell how much profit you make on the route. For example, the goal is to have profit of exactly k.

A counter reachability game



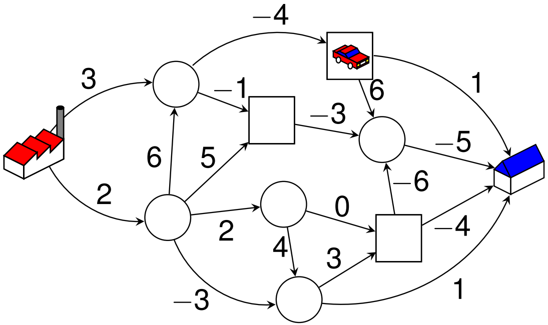
Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.

A counter reachability game



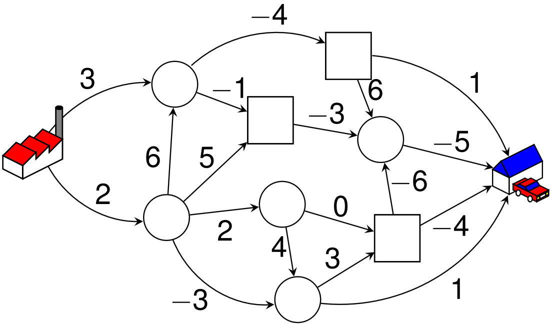
Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.

A counter reachability game



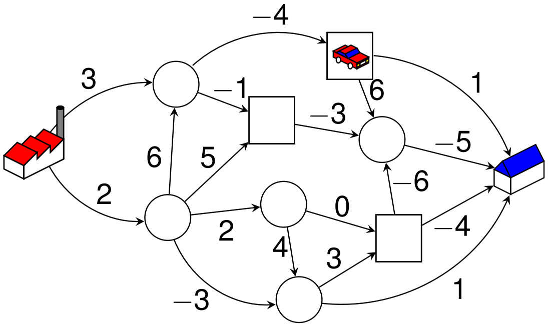
Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.

A counter reachability game



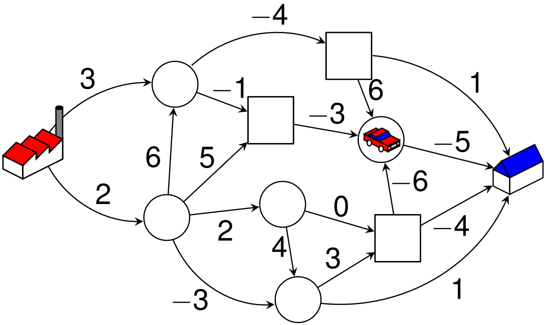
Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.

A counter reachability game



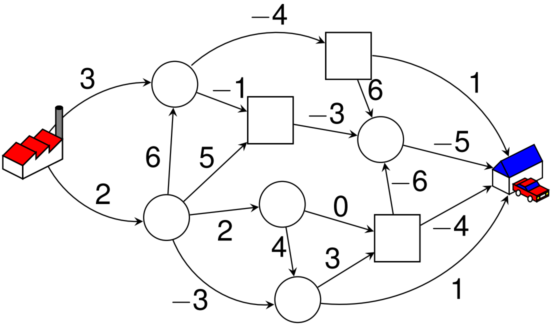
Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.

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

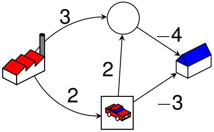


Drive home from the factory. Weights tell how much profit you make on the route. Not all routes are available – diverged traffic.

Definitions

Counter reachability games

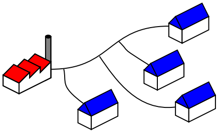
- Played on a labeled directed graph $G = (V, E)$ with edges labeled by $\mathbf{x} \in \mathbb{Z}^n$.
- Two players: Eve ($\bigcirc$) , Adam ($\square$).
- A configuration $[v, \mathbf{z}] \in V \times \mathbb{Z}^n$.
- A successor configuration is $[v', \mathbf{z} + \mathbf{z}']$, where $[v, \mathbf{z}', v'] \in E$ 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

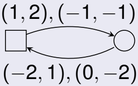
Given a graph $G = (V, E)$, initial and target configurations, $[v_0, \mathbf{z}_0]$ and $[v_f, (0, \ldots, 0)]$. Does there exist a winning strategy for Eve?

Robot games

Known for counter reachability games

One-dimensional	**EXPSPACE**-complete
Two-dimensional	Undecidable

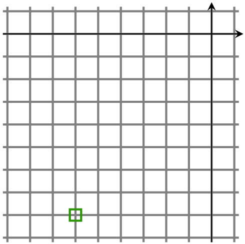
What if we have a simpler graph?



- Proposed by Doyen and Rabinovich in 2011. Claimed to be undecidable in dimension 9.
- **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 $(0, \dots, 0)$. Adam's goal is to avoid it.

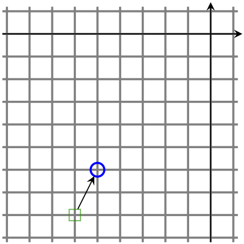


Adam's moves: $A = \{(1,2),(2,0)\}$

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.

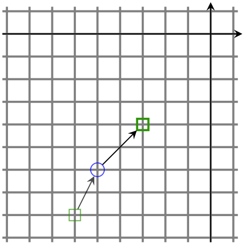


Adam's moves: $A = \{(1, 2), (2, 0)\}$

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.

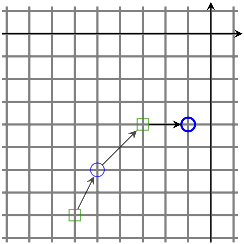


Adam's moves: $A = \{(1, 2), (2, 0)\}$

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.

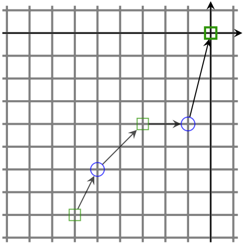


Adam's moves: $A = \{(1, 2), (2, 0)\}$

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.

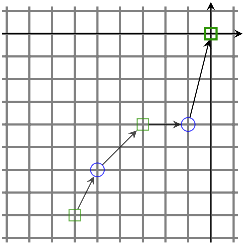


Adam's moves: $A = \{(1, 2), (2, 0)\}$

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: $A = \{(1, 2), (2, 0)\}$

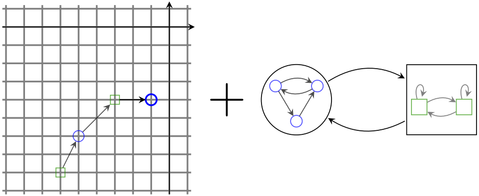
Eve's moves: $E = \{(2, 2), (1, 4)\}$

Theorem

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

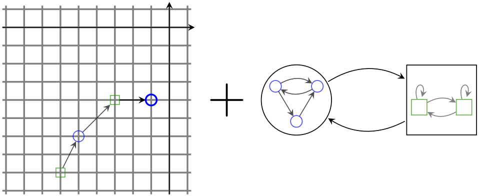
Robot games with states

- 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].



Robot games with states

- 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].



Theorem

It is undecidable which player wins in a two-dimensional robot game with states.

Robot games with states

The basis of undecidability proofs

Marvin Minsky
(1927-2016)

Deterministic two-counter Minsky machine (2CM):

- Two counters, c_1 and c_2.
- m instructions: $1 : \textsc{Ins}_1, \ldots, m : \textsc{Ins}_m$, where $\textsc{Ins}_i$ is
  - i: c_1`++; goto` k, or
  - i: c_2`++; goto` k, or
  - i: `if` c_1`=0 goto` k `else` c_1`--; goto` j, or
  - i: `if` c_2`=0 goto` k `else` c_2`--; goto` j, or
  - i: `halt`.
- The halting problem is undecidable [Minsky, 1967].

2CM example

```
1:  if c2=0 goto 2 else c2--; goto 1
2:  if c1=0 goto 5 else c1--; goto 3
3:  if c1=0 goto 7 else c1--; goto 4
4:  c2++; goto 2
5:  if c2=0 goto 2 else c2--; goto 6
6:  c1++; goto 5
7:  if c2=0 goto 8 else c2--; goto 9
8:  halt
9:  c1++; goto 9
```

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

Two-counter machines to robot games with states

- Eve simulates transitions of the machine.
- Adam verifies that zero-checks are performed correctly.

Two-counter machines to robot games with states

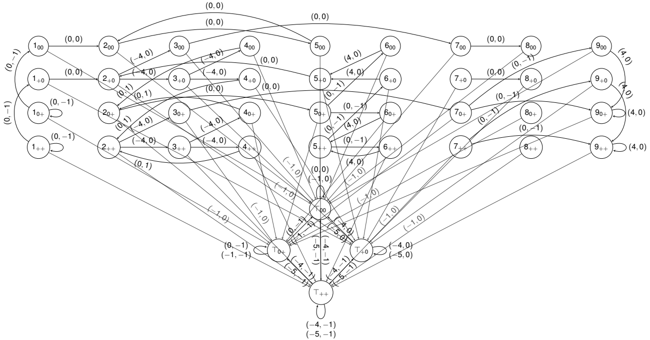
- Eve simulates transitions of the machine.
- Adam verifies that zero-checks are performed correctly.

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.

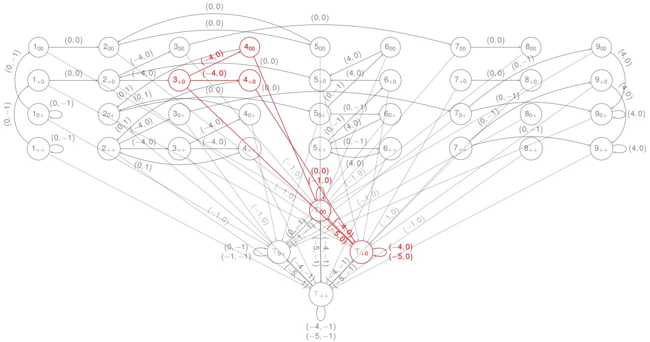
Two-counter machines to robot games with states

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

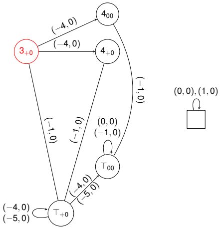


Two-counter machines to robot games with states

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:

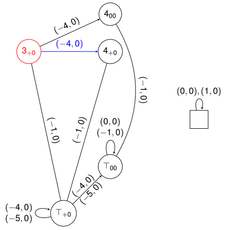
- 0-MOVE
- POSITIVITY CHECK

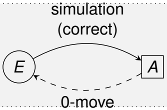
$$[3_{+0}, (8, 0)]$$

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:

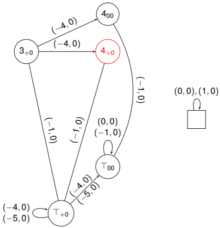
- 0-MOVE
- POSITIVITY CHECK

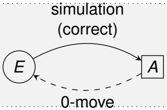
$[3_{+0}, (8,0)]$

state

counters

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

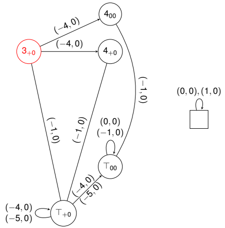
$$[3_{+0},(8,0)] \rightarrow [4_{+0},(4,0)]$$

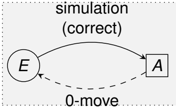
state

counters

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:

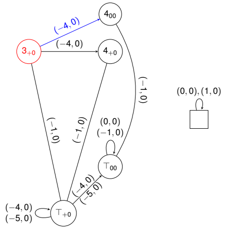
- 0-MOVE
- POSITIVITY CHECK

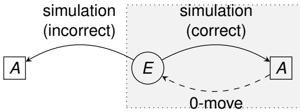
$$[3_{+0}, (8, 0)]$$

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:

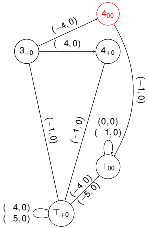
- 0-MOVE
- POSITIVITY CHECK

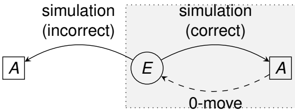
$$[3_{+0}, (8,0)]$$

state

counters

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

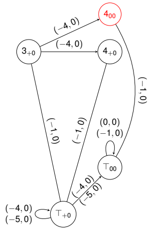
$$[3_{+0},(8,0)] \rightarrow [4_{00},(4,0)]$$

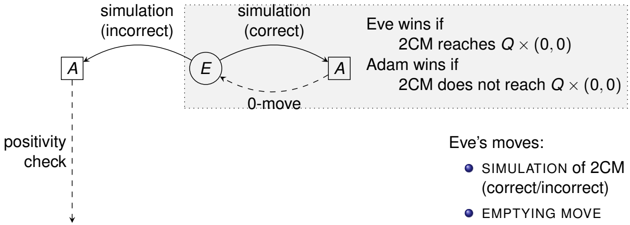
state

counters

don't match

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

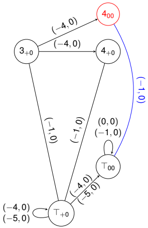
Adam's moves:

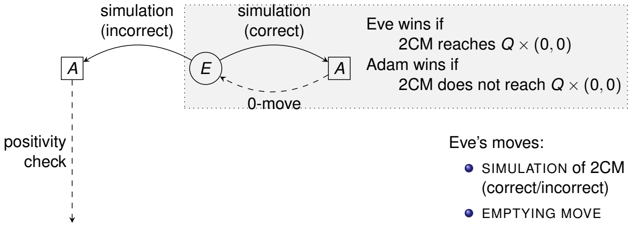
- 0-MOVE
- POSITIVITY CHECK

$$[\underbrace{3_{+0}}_{\text{state}}, \underbrace{(8,0)}_{\text{counters}}] \to [4_{00}, (4,0)] \to [4_{00}, (5,0)]$$

don't match

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

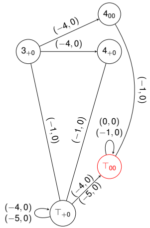
$$[3_{+0}, (8,0)] \to [4_{00}, (4,0)] \to [4_{00}, (5,0)]$$

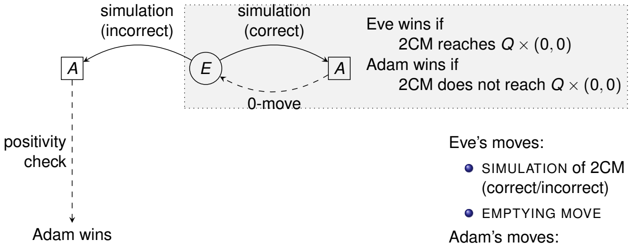
state

counters

don't match

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

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

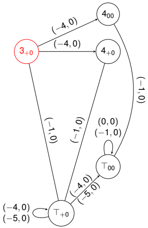
state

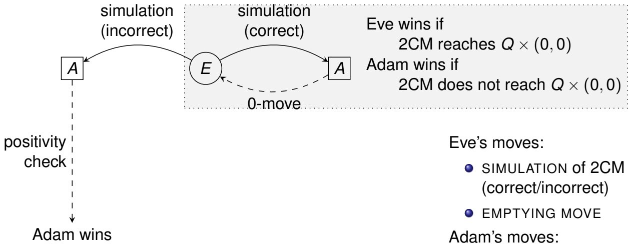
counters

don't match

Eve cannot reach (0,0)

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam's moves:

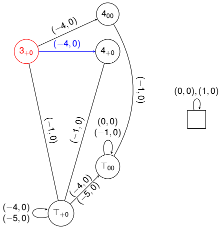
- 0-MOVE
- POSITIVITY CHECK

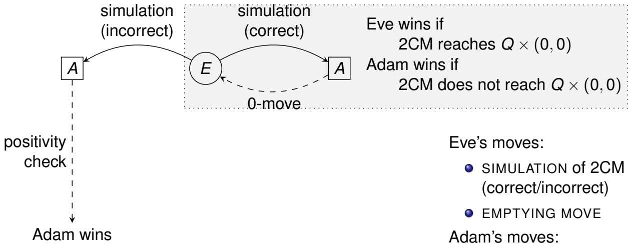
$$[3_{+0},(8,0)]$$

state

counters

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam's moves:

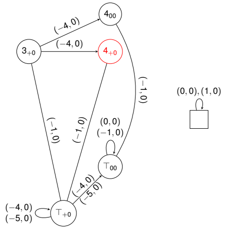
- 0-MOVE
- POSITIVITY CHECK

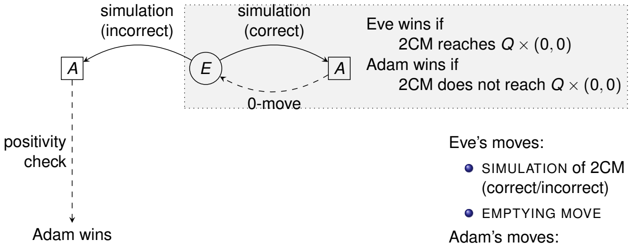
$[3_{+0},(8,0)]$

state

counters

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam's moves:

- 0-MOVE
- POSITIVITY CHECK

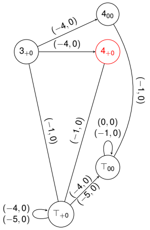
$$[3_{+0},(8,0)] \rightarrow [4_{+0},(4,0)]$$

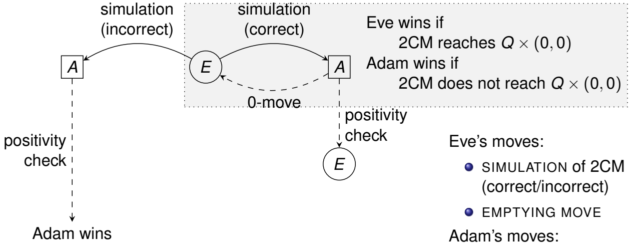
state

counters

match

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

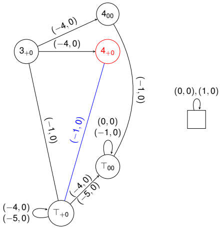
$$[3_{+0},(8,0)] \to [4_{+0},(4,0)] \to [4_{+0},(5,0)]$$

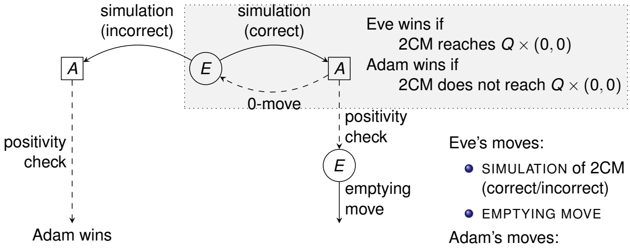
state

counters

match

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

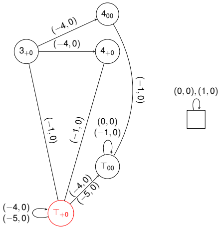
$$[3_{+0},(8,0)] \rightarrow [4_{+0},(4,0)] \rightarrow [4_{+0},(5,0)]$$

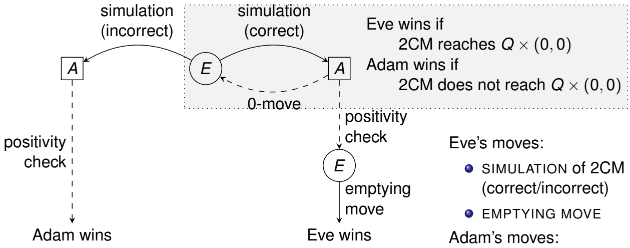
state

counters

match

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

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

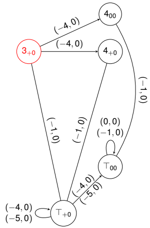
state

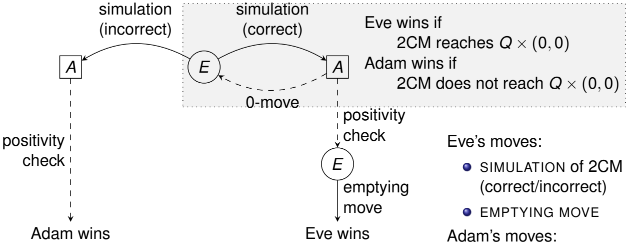
counters

match

still match

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam's moves:

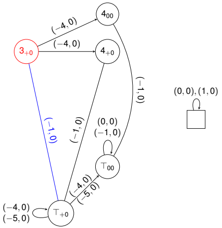
- 0-MOVE
- POSITIVITY CHECK

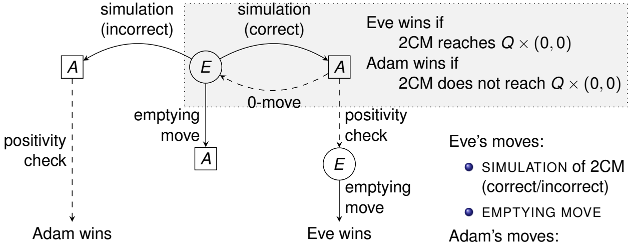
$[3_{+0}, (8, 0)]$

state

counters

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam's moves:

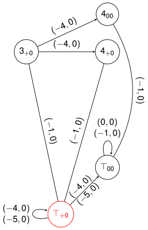
- 0-MOVE
- POSITIVITY CHECK

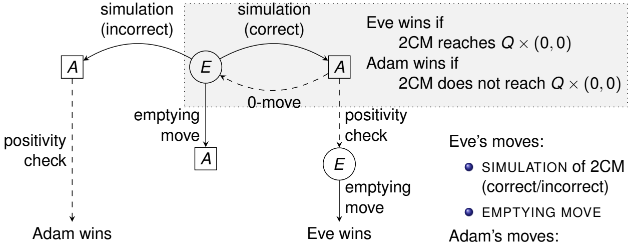
$[3_{+0},(8,0)]$

state

counters

Two-counter machines to robot games with states





Eve's moves:

- SIMULATION of 2CM (correct/incorrect)
- EMPTYING MOVE

Adam's moves:

- 0-MOVE
- POSITIVITY CHECK

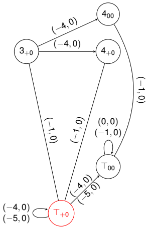
$$[3_{+0},(8,0)] \to [\top_{+0},(7,0)]$$

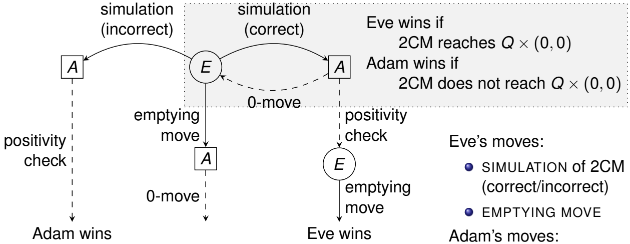
state

counters

not $0 \pmod 4$

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

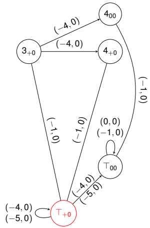
$$[3_{+0},(8,0)] \to [\top_{+0},(7,0)] \to [\top_{+0},(7,0)]$$

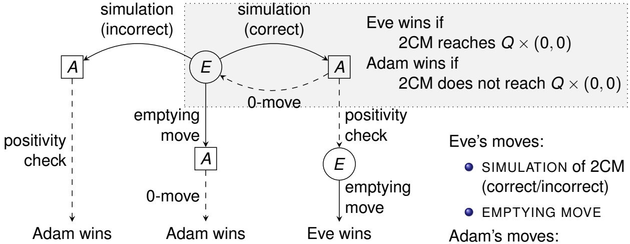
state

counters

not $0 \pmod 4$

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

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

state

counters

not 0 $\pmod 4$

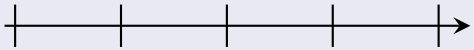
Robot games

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.

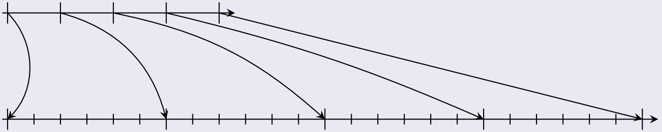
Robot games with states to robot games

- 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.



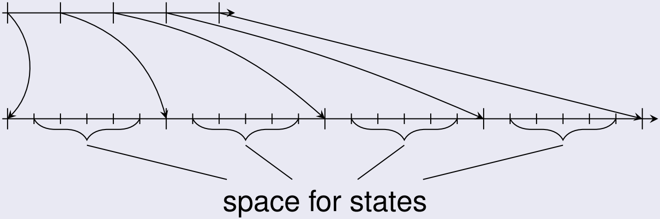
Robot games with states to robot games

- 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.



Removing the states from robot games with states

A move in 2RGS:

$$i \xrightarrow{(x,y)} j$$

The corresponding move in 2RG:

$(x, yN - 2^i + 2^j)$, 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:

$(x, yN - 2^i + 2^j)$, 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:

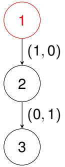
$$i \xrightarrow{(x,y)} j$$

The corresponding move in 2RG:

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

- Too simple: Several wrong moves can result in a right one.
- Even more space is needed to make sure that wrong moves do not affect the rest of the computation.

Robot games with states to robot games



Eve's moves:

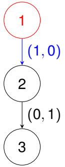
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

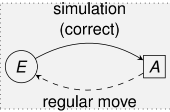
Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$(1, \overbrace{0 \cdot 4 \cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0 \cdot 8^{9} + 0 \cdot 8^{8} + 0 \cdot 8^{7} + 0 \cdot 8^{6} + 0 \cdot 8^{5} + 0 \cdot 8^{4}}^{\text{emptying states}} + \overbrace{0 \cdot 8^{3} + 0 \cdot 8^{2} + 1 \cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^{0}}^{\top_{00}})$$

Robot games with states to robot games





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

Eve's moves:

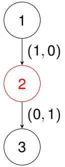
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

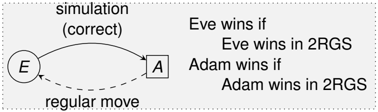
Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$(\overbrace{1, 0 \cdot 4 \cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4}^{\text{emptying states}} + \overbrace{0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^0}^{\top_{00}})$$

Robot games with states to robot games





Eve's moves:

- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:

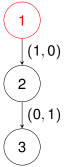
- REGULAR MOVE
- STATE-CHECK

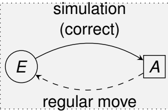
$$(\overbrace{1, 0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{0\cdot 8^{3} + \boxed{0\cdot 8^{2}} + \boxed{1\cdot 8^{1}}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top_{00}})$$

$$\downarrow (1, -8^{1} + 8^{2})$$

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

Robot games with states to robot games





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

Eve's moves:

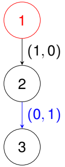
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

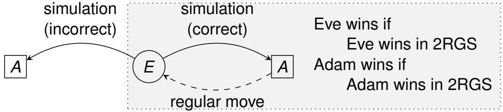
Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$\overbrace{(1, 0 \cdot 4 \cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4}^{\text{emptying states}} + \overbrace{0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^0}^{\top_{00}})$$

Robot games with states to robot games





Eve's moves:

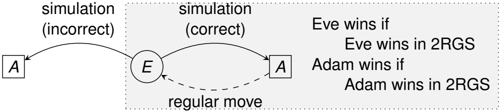
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

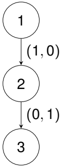
Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$(\overbrace{1, 0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{0\cdot 8^{3} + 0\cdot 8^{2} + 1\cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top 00})$$

Robot games with states to robot games





Eve's moves:

- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:

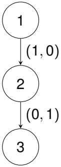
- REGULAR MOVE
- STATE-CHECK

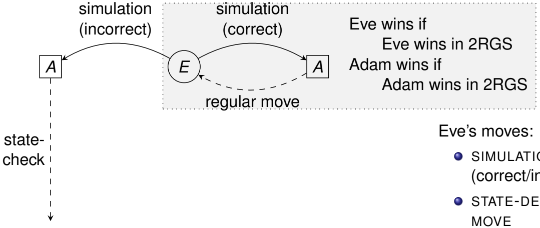
$$\Big(\overbrace{1, 0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{\boxed{0\cdot 8^{3} + 0\cdot 8^{2}} + 1\cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top_{00}}\Big)$$

$$\downarrow (0, 1\cdot 4\cdot 8^{10} - 8^{2} + 8^{3})$$

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

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

$$(1, \overbrace{0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{0\cdot 8^{3} + 0\cdot 8^{2} + 1\cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top_{00}})$$

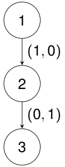
$$\downarrow (0, 1\cdot 4\cdot 8^{10} - 8^{2} + 8^{3})$$

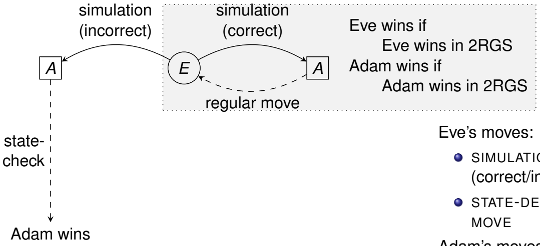
$$(1, 1\cdot \boxed{4\cdot 8^{10} + 0\cdot 8^{9}} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4} + 1\cdot 8^{3} - 1\cdot 8^{2} + 1\cdot 8^{1} - 1\cdot 8^{0})$$

$$\downarrow (0, -8^{10} - 5\cdot 8^{9})$$

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

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

$$(1, \overbrace{0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{0\cdot 8^{3} + 0\cdot 8^{2} + 1\cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top_{00}})$$

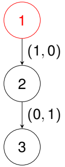
$$\downarrow (0, 1\cdot 4\cdot 8^{10} - 8^{2} + 8^{3})$$

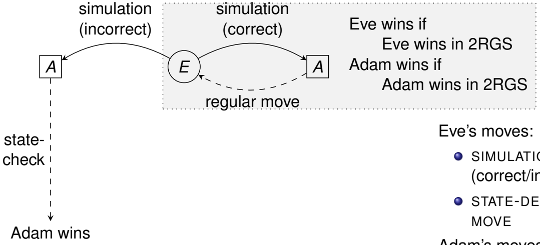
$$(1, 1\cdot \boxed{4\cdot 8^{10} + 0\cdot 8^{9}} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4} + 1\cdot 8^{3} - 1\cdot 8^{2} + 1\cdot 8^{1} - 1\cdot 8^{0})$$

$$\downarrow (0, -8^{10} - 5\cdot 8^{9})$$

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

Robot games with states to robot games





Eve's moves:

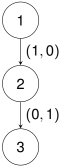
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

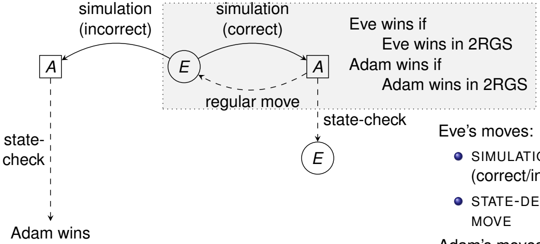
Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$\overbrace{(1, 0 \cdot 4 \cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0 \cdot 8^{9} + 0 \cdot 8^{8} + 0 \cdot 8^{7} + 0 \cdot 8^{6} + 0 \cdot 8^{5} + 0 \cdot 8^{4}}^{\text{emptying states}} + \overbrace{0 \cdot 8^{3} + 0 \cdot 8^{2} + 1 \cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^{0})}^{\top_{00}}$$

Robot games with states to robot games





Eve's moves:

- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:

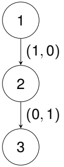
- REGULAR MOVE
- STATE-CHECK

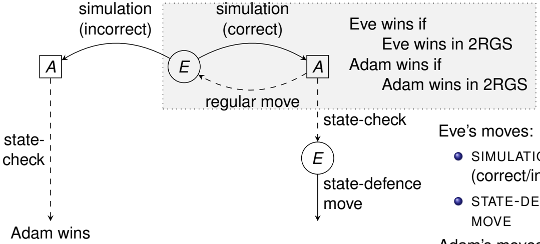
$$(1, \overbrace{\boxed{0 \cdot 4 \cdot 8^{10} + 0 \cdot 8^{9}}}^{\text{2RGS counters}} \overbrace{+ 0 \cdot 8^{8} + 0 \cdot 8^{7} + 0 \cdot 8^{6} + 0 \cdot 8^{5} + 0 \cdot 8^{4}}^{\text{emptying states}} \overbrace{+ 0 \cdot 8^{3} + 0 \cdot 8^{2} + 1 \cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^{0}}^{\top_{00}})$$

$$\downarrow (0, -8^{10} - 5 \cdot 8^{9})$$

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

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

$$(1, \overbrace{0 \cdot 4 \cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4}^{\text{emptying states}} + \overbrace{0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^0}^{\top_{00}})$$

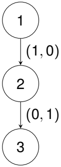
$$\downarrow (0, -8^{10} - 5 \cdot 8^9)$$

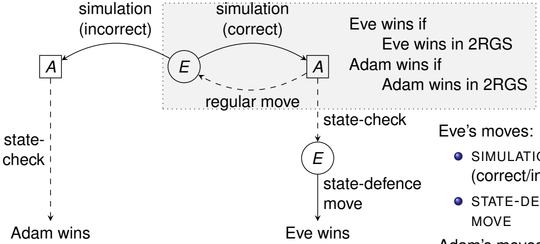
$$(1, \boxed{-1 \cdot 8^{10} - 5 \cdot 8^9} + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + \boxed{0 \cdot 8^5} + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + \boxed{1 \cdot 8^1} - 1 \cdot 8^0)$$

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

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

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

$$\overbrace{(1, 0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{0\cdot 8^{3} + 0\cdot 8^{2} + 1\cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top_{00}})$$

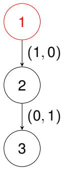
$$\downarrow (0, -8^{10} - 5\cdot 8^{9})$$

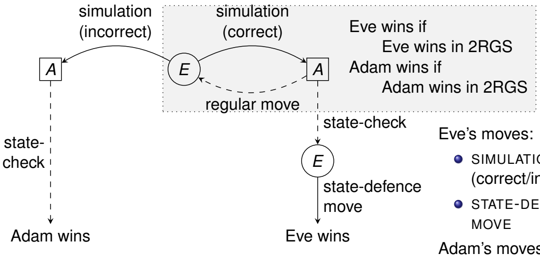
$$(1, \boxed{-1\cdot 8^{10} - 5\cdot 8^{9}} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + \boxed{0\cdot 8^{5}} + 0\cdot 8^{4} + 0\cdot 8^{3} + 0\cdot 8^{2} + \boxed{1\cdot 8^{1}} - 1\cdot 8^{0})$$

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

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

Robot games with states to robot games





Eve's moves:

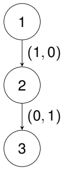
- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

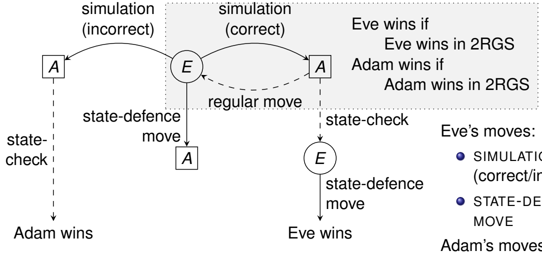
Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$(1, \overbrace{0\cdot 4\cdot 8^{10}}^{\text{2RGS counters}} + \overbrace{0\cdot 8^{9} + 0\cdot 8^{8} + 0\cdot 8^{7} + 0\cdot 8^{6} + 0\cdot 8^{5} + 0\cdot 8^{4}}^{\text{emptying states}} + \overbrace{0\cdot 8^{3} + 0\cdot 8^{2} + 1\cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1\cdot 8^{0}}^{\top_{00}})$$

Robot games with states to robot games





Eve's moves:

- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:

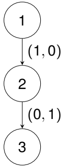
- REGULAR MOVE
- STATE-CHECK

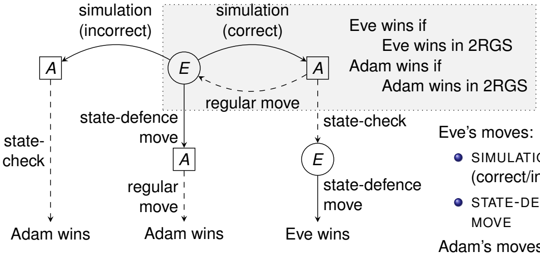
$$\overbrace{(1, 0 \cdot 4 \cdot 8^{10}}^{\text{2RGS counters}} \overbrace{+ 0 \cdot 8^{9} + 0 \cdot 8^{8} + 0 \cdot 8^{7} + 0 \cdot 8^{6} + 0 \cdot 8^{5} + 0 \cdot 8^{4}}^{\text{emptying states}} \overbrace{+ 0 \cdot 8^{3} + 0 \cdot 8^{2} + 1 \cdot 8^{1}}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^{0}}^{\top 00})$$

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

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

Robot games with states to robot games





Eve's moves:

- SIMULATION of 2RGS (correct/incorrect)
- STATE-DEFENCE MOVE

Adam's moves:

- REGULAR MOVE
- STATE-CHECK

$$(1, 0 \cdot \overbrace{\boxed{4 \cdot 8^{10} + 0 \cdot 8^{9}}}^{\text{2RGS counters}} \overbrace{+ 0 \cdot 8^{8} + 0 \cdot 8^{7} + 0 \cdot 8^{6} + \boxed{0 \cdot 8^{5}} + 0 \cdot 8^{4}}^{\text{emptying states}} \overbrace{+ 0 \cdot 8^{3} + 0 \cdot 8^{2} + \boxed{1 \cdot 8^{1}}}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^{0}}^{\top_{00}})$$

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

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

Conclusion

Summary

Theorem

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

Game	Dimension		
	1	2	≥ 3
counter reachability games	**EXPSPACE**-complete	U	–
robot games with states	**EXPSPACE**-complete	U	–
robot games	**EXPTIME**-complete	?	U

Summary

Theorem

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

Game	Dimension		
	1	2	≥ 3
counter reachability games	**EXPSPACE**-complete	U	–
robot games with states	**EXPSPACE**-complete	U	–
robot games	**EXPTIME**-complete	U	–

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!