

On Decidability and Complexity of Low-dimensional Robot Games[☆]

R. Niskanen^{a,1,*}, I. Potapov^{b,2}, J. Reichert^c

^a*Department of Computer Science, University of Oxford, Wolfson Building, Oxford, OX1 3QD, UK*

^b*Department of Computer Science, University of Liverpool, Ashton Building, Liverpool, L69 3BX, UK*

^c*LSV, ENS Cachan, France*

Abstract

A robot game, also known as \mathbb{Z} -VAS game, is a two-player vector addition game played on the integer lattice \mathbb{Z}^n , where one of the players, Adam, aims to avoid the origin while the other player, Eve, aims to reach the origin. The problem is to decide whether or not Eve has a winning strategy. In this paper we prove undecidability of the two-dimensional robot game closing the gap between undecidable and decidable cases. We also prove that deciding the winner in a robot game with states in dimension one is EXPSPACE-complete and study a subclass of robot games where deciding the winner is in EXPTIME.

Keywords: reachability games, vector addition game, decidability, winning strategy, integer vector addition system

1. Introduction

In the modern world the reliability of a software code and verification of the correct functionality of complex technological devices require the analysis of various interactive processes and open systems, where it is important to take into account the effects of uncontrollable adversaries, such as environment or malicious users. Two-player computational games provide a powerful framework for problems related to verification and refinement of reactive systems [3], and have deep connections with automata theory and logic [4, 5]. Infinite-state games can be classified according to the winning conditions, such as parity [6], energy [7], counter reachability, or a combination of two or more winning conditions [8]. The extensions of classical reachability problems to game schemes, studied in different contexts and settings, have recently garnered considerable interest [9, 6, 10, 11, 8, 7, 12].

[☆]This paper is an extended version of [1] and [2].

*Corresponding author

Email addresses: reino.niskanen@cs.ox.ac.uk (R. Niskanen), potapov@liverpool.ac.uk (I. Potapov), reichert@crans.org (J. Reichert)

¹The research was done while the author was at the University of Liverpool.

²The author was partially supported by EPSRC grant “Reachability problems for words, matrices and maps” (EP/M00077X/1)

In this paper we study two-player games where the main problem is to decide which of the players wins based on a given set of eligible moves, a computational environment and reachability objectives. Following early results for games on VASS (Vector Addition Systems with States)³ [13, 11], Doyen and Rabinovich formulated an open problem about the simplest version of games (*robot games*) for which the decidability was unknown [14]. Robot games are two-player games played by updating a vector of n integer counters. Each of the players, called Adam and Eve, has a finite set of vectors in \mathbb{Z}^n . A play starts from a given initial vector $\mathbf{x}_0 \in \mathbb{Z}^n$, and proceeds in rounds. During each round, first Adam adds a vector from his set, followed by Eve doing the same. Eve wins when, after her turn, the vector is the zero vector. A simple example of the game is illustrated in Figure 1. A generalisation of VASS, where the condition on positivity of the counters of VASS is relaxed to also allow negative values, known as integer VASS or \mathbb{Z} -VASS has been studied before [15, 16, 17, 18]. Due to this connection, robot games are sometimes also called games on \mathbb{Z} -VAS.

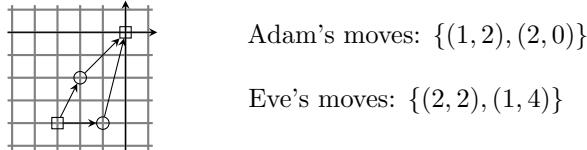


Figure 1: An example of a robot game. Eve (circle) has a winning strategy from the configurations in $\{(-3a, -4a) \mid a \in \mathbb{N}\}$.

We say that Eve has a winning strategy if she eventually can reach the zero vector regardless of the moves Adam plays. As a consequence of [19], robot games are determined, that is, Adam has a winning strategy if Eve does not. Thus a winning strategy gives a way for a player to win, regardless of the way the opponent plays. Previously, it has been proved that deciding the winner in one-dimensional robot games, where integers are given in binary, is EXPTIME-complete [20].

In the first part of the paper we consider the open problem of deciding the winner of robot games for dimension $n = 2$ and show that it is undecidable to check which of the players has a winning strategy in a two-dimensional robot game, i.e., in a very restricted fragment of counter reachability games with stateless players playing in integer grid \mathbb{Z}^2 .

The basis of proofs are 2-counter Minsky machines (2CM) for which the halting problem is undecidable. For a 2-counter machine, we construct a game where Eve has to simulate the machine and Adam verifies that Eve does not cheat. The intuition is that the counters of the machine are multiplied by constants and represented by two-dimensional vectors. Additionally, the states of the machine are encoded in the least significant digits of the vectors. We analyse all the possible deviations from simulating the counter machine and show that the opponent has a winning strategy in that case. The biggest challenge is to ensure that all possible ways to cheat can be caught without introducing new ways to cheat for the other player.

We prove the main theorem by considering the undecidable problem of determining whether a 2CM \mathcal{M} reaches a configuration where both counters are zero. In Section 3,

³A game is played on a graph with states of player 1 and states of player 2, with \mathbb{N}^2 as the vector space.

we construct a robot game with states that follows the computation of \mathcal{M} . To simulate zero checks present in 2CM, Adam has a move allowing him to check whether a counter is positive or not. This check leads, deterministically, either to Adam’s victory with a correct guess or to his loss otherwise. In the fourth section, we map the states and state transitions into integers and embed them into the least significant digits in vectors of a two-dimensional robot game. Our proof uses two successive reductions making the proof shorter and more intuitive in contrast to a direct reduction from 2CM that would lead to a longer proof with significantly more cases to consider.

Known results on robot games in different dimensions are summarised in Table 1. Apart from the solution of the open problem, the main contribution of the first part is a collection of new, original encodings and constructions that allow simulating zero-checks and state space of a universal machine within a minimalistic two-dimensional system of two non-deterministic stateless players.

Dimension	1	2	3
	EXPTIME-complete [20]	undecidable	undecidable [21]

Table 1: The results on complexity of deciding whether Eve has a winning strategy in robot games. Our result is in bold.

In the second part, we consider an extension of robot games, where both players have internal control states, called *robot games with states* (RGS). We prove that robot games with states are undecidable starting from dimension two. Unlike counter reachability games [12] and robot games [20], one-dimensional robot games with states have not been studied before. Our main result is to prove that robot games with states in dimension one are EXPSPACE-complete by presenting a mutual reduction between robot games with states and counter reachability games. Note that this is not obvious as the games have essential differences. In a counter reachability game, since the game is played on a graph, a choice of a player, say Eve, affects from which state Adam moves next. In fact, it is not guaranteed that Adam will move at all, as it is possible for Eve to move only between her states. On the other hand, in robot games with states, the next state of a player is determined only by his or her previous move.

In order to show EXPSPACE-hardness, we construct a one-dimensional robot game with states that can simulate any one-dimensional counter reachability game such that Eve has a winning strategy in the RGS if and only if she has a winning strategy in the counter reachability game. The idea is for Eve to simulate the whole graph of the counter reachability game and for Adam to verify that Eve is simulating correctly. In the constructed robot game with states Eve has $n + 1$ states, where the counter reachability game has n vertices, and Adam has only one state. Then to show completeness, we transform a robot game with states into a counter reachability game such that Eve has a winning strategy in the counter reachability game if and only if she has a winning strategy in the robot game with states. The construction is relatively simple and involves storing information on the state of a player into the states of the opponent.

Seeing how adding states for Eve increases complexity from EXPTIME of robot games to EXPSPACE of robot games with states, we consider a state structure of Adam that does not increase the complexity of the game. We show that deciding the winner in one-dimensional robot game with states, where Eve is stateless and Adam’s states are *flat*, is

in EXPTIME. Flat automata have been studied in various contexts [22, 23, 24, 25, 26] and have been shown to be a fruitful tool in verification of counter automata. Flat automata is a subclass of automata where the automaton does not have nested loops. This particular structure allows us to break a robot game with states into several stateless games, that can be solved in EXPTIME. The main challenge is in connecting these separate games. As Adam's underlying state structure is flat, there are only finitely many transitions from one game to another. This fact together with the particular structure of winning sets constructed by the algorithm for a stateless game of [20] provide us with necessary tools to decide the winner in EXPTIME.

Adam \ Eve	states	flat states	stateless
states	EXPSpace (Lemma 17)	?	?
flat states	—	?	EXPTIME (Theorem 8)
stateless	EXPSpace-hard (Theorem 3)	?	EXPTIME-c. [20]

Table 2: Complexity of checking for the existence of a winning strategy for Eve in different variants of one-dimensional robot games with states. The propagation of upper bounds is depicted with double arrows and of lower bounds with dotted arrows.

In Table 2 is a summary of complexity results on one-dimensional robot games with states according to the state structure of each player. Most of the variants do not have tight complexity and are between EXPSpace and EXPTIME-hard. Only robot games (i.e., both players are stateless) and robot games where Adam has flat states are EXPTIME-complete, while the robot games where Eve has states are EXPSpace-complete, regardless on the state structure of Adam.

The paper is organized as follows. In the next section we introduce the notation and definitions used throughout the paper. In Section 3, we construct a two-dimensional robot game with states that follows the computation of \mathcal{M} and show that 2RGS are undecidable. In the fourth section, we map the states and state transitions into integers and embed them into the least significant digits in vectors of a two-dimensional robot game, showing that also the stateless two-dimensional robot games are undecidable. In the fifth section, we prove that deciding the winner in one-dimensional robot games with states is EXPSpace-complete. In Section 6, we consider flat robot games with states and prove that, in dimension one, deciding the winner is EXPTIME-complete.

2. Notation and definitions

We denote the sets of integers, non-positive and non-negative integers (that is, natural numbers) by \mathbb{Z} , \mathbb{Z}^- and \mathbb{Z}^+ respectively. By 0_n we denote the n -dimensional zero vector. An open interval (a, b) is a subset of \mathbb{Z} containing all the integers larger than a and smaller than b . A closed interval $[a, b]$ is $(a, b) \cup \{a, b\}$ and half-open intervals are defined

similarly. Let $X \subseteq \mathbb{Z}$. By $X + d$ and dX , where $d \in \mathbb{Z}$, we denote the sets $\{x + d \mid x \in X\}$ and $\{dx \mid x \in X\}$.

A n -dimensional *counter reachability game* (n CRG) consists of a directed graph $G = (V, F)$, where the set of vertices is partitioned into two parts, V_E and V_A , each edge $e \in F \subseteq V \times \mathbb{Z}^n \times V$ is labelled with vectors in \mathbb{Z}^n . A *configuration* of the game is $[v, \mathbf{x}]$, a successive configuration is $[v', \mathbf{x} + \mathbf{x}']$, where an edge $(v, \mathbf{x}', v') \in F$ is chosen by player 1 if $v \in V_E$ or by player 2 if $v \in V_A$. A *play* is a sequence of successive configurations. The goal of the first player, called *Eve*, is to reach the *final configuration* $[v_f, 0_n]$ for some $v_f \in V$ from a given *initial configuration* $[v_0, \mathbf{x}_0]$, while the goal of the second player, called *Adam*, is to keep Eve from reaching $[v_f, 0_n]$. A *strategy* for a player is a function that maps a configuration to an edge that can be applied. We say that Eve has a *winning strategy* if she can reach the final configuration regardless of the strategies of Adam. On the other hand, we say that Adam has a winning strategy if Eve does not have a winning strategy. In the figures we use \circlearrowleft for Eve's states and \square for Adam's states (diamonds represent arbitrary vertices).

A n -dimensional *robot game* (n RG) [14] is a special case of the counter reachability games, where graph consists of only two vertices, v_0 of Adam and v of Eve, and edges are of the form (v_0, \mathbf{x}, v) and (v, \mathbf{x}, v_0) (i.e., there are no self-loops). The goal of the game is the configuration $[v_0, 0_n]$. That is, a robot game consists of two players, *Eve* and *Adam*, having a set of vectors E, A over \mathbb{Z}^n , respectively, and an *initial vector* \mathbf{x}_0 . Starting from \mathbf{x}_0 players add a vector from their respective sets to the current configuration of the game in turns. As in counter reachability game, Eve tries to reach the origin while Adam tries to keep Eve from reaching the origin. The decision problem concerning robot games is, for a given robot game (A, E) and \mathbf{x}_0 , to decide whether Eve has a winning strategy to reach 0_n from \mathbf{x}_0 .

An extension of robot games where players have control states is called *robot games with states* (RGS). A n RGS consists of (A, E) where A is a finite subset of $Q_A \times \mathbb{Z}^n \times Q_A$ that Adam can apply during his turn and E is a finite subset of $Q_E \times \mathbb{Z}^n \times Q_E$ of Eve, and an initial configuration $[s_0, t_0, \mathbf{x}_0] \in Q_E \times Q_A \times \mathbb{Z}^n$. The configuration is now a triple $[s, t, \mathbf{v}]$ consisting of Eve's control state s , Adam's control state t and a counter vector $\mathbf{v} \in \mathbb{Z}^n$. Eve updates her control state when she makes a move: in the configuration $[s, t, \mathbf{v}]$, for any vector \mathbf{v} , only moves of the form (s, \mathbf{x}, s') are enabled, and with one such move the new configuration is $[s', t, \mathbf{v} + \mathbf{x}]$. Similarly Adam updates his control state when he makes a move. Eve wins if, and only if, after her turn, the configuration is $[s, t, 0_n]$ for some $s \in F \subseteq Q_E$ and any $t \in Q_A$. The decision problem associated with robot games with states asks whether Eve has a winning strategy from a given configuration.

In order to indicate whose turn it is in the configuration $[s, t, \mathbf{v}]$, we put a dot above s if it is Eve's turn, or above t if it is Adam's turn. That is, the respective configurations are $[s, t, \mathbf{v}]$ and $[s, \dot{t}, \mathbf{v}]$. In the figures, the dot is placed inside the state (e.g., \square if it is Adam's turn).

Flat robot games with states (FRGS) is a subclass of the RGS where Eve is stateless, that is, all the moves of Eve are of form (s, z, s) , and Adam's states are flat, i.e., without nested loops. In other words, we have an ordering of states of Adam $\{t_0, \dots, t_k\}$ such that $(t_i, z, t_j) \in A$ only if $i \leq j$. Note that, unlike the usual definition of flat systems, we allow several self-loops for a state.

A *Minsky machine*, introduced in [27], is a simple computation model that is crucial in our proof. A *deterministic two-counter Minsky machine* (2CM) is a pair (Q, T) , where

Q is a finite set of states and $T \subseteq Q \times \{c_{i++}, c_{i--}, c_{i==0} \mid i = 1, 2\} \times Q$ is a finite set of labelled transitions to increment, decrement or test for zero one of the counters. In a deterministic two-counter Minsky machine, the set Q contains an initial state s_0 and a sink state \perp , such that there is no outgoing transition from \perp . Moreover, from all $s \in Q \setminus \{\perp\}$, either there is only one outgoing transition with the label c_{1++} or c_{2++} , or there are exactly two outgoing transitions with respective labels c_{1--} and $c_{1==0}$, or c_{2--} and $c_{2==0}$. A configuration of a 2CM is a pair $(s, (y, z)) \in Q \times (\mathbb{Z}^+)^2$, representing a state and a pair of counter values. The run of a 2CM is a finite or infinite sequence of configurations that starts from $(s_0, (0, 0))$ and follows the transitions of the machine incrementing and decrementing the counters according to the labels. As usual, a transition with a label $c_{i==0}$ can only be taken when the counter i is zero and a transition with a label c_{i--} can only be taken when the counter i is positive.

Note that there is only one possible run in a deterministic two-counter Minsky machine. Indeed, when there are two outgoing transitions, only one of them can be executed, depending on the value of the counter that the transitions update or test for zero. The halting problem of 2CM is to decide, given a 2CM, whether the run reaches a configuration with state \perp , in other words whether the run halts. This problem is known to be undecidable for deterministic two-counter machines [27]. Another well-known undecidable problem for 2CM is whether the machine halts with both counters zero. We are interested in more general question: whether in the run of a 2CM both counters are zero at some point. This problem is undecidable and the proof follows from the halting problem by modifying a 2CM to ensure that both counters are zero only in the halting state; see [21] for a proof.

Theorem 1. *Let (Q, T) be a deterministic two-counter machine. It is undecidable whether in the run of (Q, T) , a configuration in $Q \times \{(0, 0)\} \setminus \{(s_0, (0, 0))\}$ appears.*

We can assume that the first move of a 2CM is an increment of either c_1 or c_2 . Indeed, otherwise the problem is trivial as the second configuration is in $Q \times \{(0, 0)\}$.

3. Robot games with states in two dimensions

In this section we prove that the decision problem for robot games with states is undecidable. We show that for each two-counter machine, there exists a corresponding robot game with states where Eve has a winning strategy if and only if the machine reaches a configuration where both counters are zero. To simulate zero checks present in two-counter machines, Adam has a move allowing him to check whether a counter is positive or not.

Theorem 2. *Let (Q, T) be a two-counter machine. There exists a effectively constructable two-dimensional robot game with states (A, E) where Eve has a winning strategy if and only if (Q, T) reaches a configuration in $Q \times \{(0, 0)\}$.*

The idea is that in the robot game with states, Eve simulates the computation of the 2CM while Adam does not interfere with the computation. If one of the players deviates from the computation, the opponent has a winning strategy from that point on.

Essentially, there are four ways the game can progress. These ways are depicted in the Figure 2. Three of the outcomes have a predetermined winner which does not depend

on the 2CM. In the last case where Eve correctly simulates the 2CM and Adam does not interfere (plays only a 0-MOVE), the winner depends on whether the 2CM reaches $(s, (0, 0))$ for some $s \in Q$ or not.

- If Eve's move corresponds to the SIMULATION of the 2CM and Adam replies with a 0-MOVE (a move that does not modify the counters), then iteratively applying only this turn-based interaction, Eve wins if and only if the 2CM reaches $(s, (0, 0))$ for some $s \in Q$ (Lemma 3).
- If Eve's move incorrectly simulates the 2CM, then Adam has a winning strategy from this moment on, starting with a POSITIVITY CHECK that makes Eve's target unreachable (Lemma 4).
- On the other hand, if Adam plays his POSITIVITY CHECK following a correct simulating move of Eve, then Eve has a winning strategy from this moment on, starting with an EMPTYING MOVE allowing Eve to empty both counters and reach $(0, 0)$ (Lemma 5).
- Finally, if Eve plays an EMPTYING MOVE instead of a SIMULATING MOVE, in that case Adam has a winning strategy starting by playing his 0-MOVE (Lemma 6).

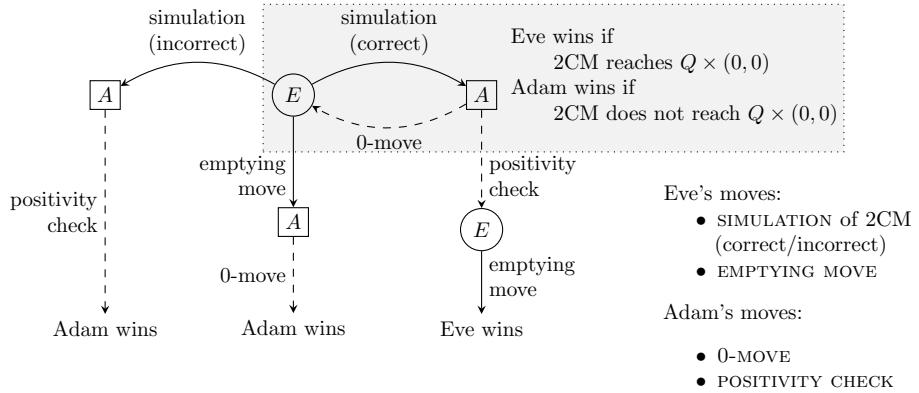


Figure 2: Progress of 2RGS. Here solid arrows are moves of Eve and dashed arrows are moves of Adam.

Before presenting the detailed constructions of Eve's and Adam's state spaces, we consider a simple modification to a 2CM, making it non-deterministic. For any 2CM (Q, T) , we construct a 2CM (Q', T') where Q' is Q with additional information on positivity of the both counters and T' is like T with guards ensuring that the extra information in states of Q' correspond to the actual values of the counters. We denote the states of Q' by s_{ab} where $s \in Q$ and $a, b \in \{0, +\}$ are flags indicating whether the value of a counter is positive or equal to 0, i.e., a (b) is $+$ if the first (second) counter is positive or 0 if the counter is zero. The transition set T' consists of the following sets

$$\begin{aligned} &\{(s_{ab}, c_{1++}, t_{+b}) \mid (s, c_{1++}, t) \in T, a, b \in \{0, +\}\}, \{(s_{ab}, c_{2++}, t_{a+}) \mid (s, c_{2++}, t) \in T, a, b \in \{0, +\}\}, \\ &\{(s_{+b}, c_{1--}, t_{ab}) \mid (s, c_{1--}, t) \in T, a, b \in \{0, +\}\}, \{(s_{a+}, c_{2--}, t_{ab}) \mid (s, c_{2--}, t) \in T, a, b \in \{0, +\}\}, \\ &\{(s_{0b}, c_{1==0}, t_{0b}) \mid (s, c_{1==0}, t) \in T, b \in \{0, +\}\}, \{(s_{a0}, c_{2==0}, t_{a0}) \mid (s, c_{2==0}, t) \in T, a \in \{0, +\}\}. \end{aligned}$$

Now, after decrementing counters from a state with + flag, a state will be changed to a state with + or 0 flag depending on the current counter value.

counter value	flag	flag	
$c_i > 1$	+	\rightarrow	+
$c_i > 1$	+	\rightarrow	0
$c_i = 1$	+	\rightarrow	+
$c_i = 1$	+	\rightarrow	0

At the moment we assume that the machine moves to a state with the correct flag (correct simulation) and does not move to incorrect flag (incorrect simulation). Later in the robot game with states, Adam will act as guards (i.e., checks whether $c_i > 1$ or $c_i = 1$) using his POSITIVITY CHECK if Eve picks a wrong transition resulting in a state with the wrong flag.

Now we present the moves of the players. Eve's states are the states of Q' , corresponding to the simulation of the 2CM, together with emptying states $\{\top_{00}, \top_{+0}, \top_{0+}, \top_{++}\}$, associated with EMPTYING MOVES. The moves of Eve correspond to transitions in T' where incrementing and decrementing of the first counter is by 4 rather than by 1. We call these moves SIMULATING MOVES.

Transition with c_1	Eve's move	Transition with c_2	Eve's move
(s, c_1++, t)	$(s, (4, 0), t)$	(s, c_2++, t)	$(s, (0, 1), t)$
(s, c_1--, t)	$(s, (-4, 0), t)$	(s, c_2--, t)	$(s, (0, -1), t)$
$(s, c_1==0, t)$	$(s, (0, 0), t)$	$(s, c_2==0, t)$	$(s, (0, 0), t)$

The other type of moves, EMPTYING MOVES, are related to the new states and are used to empty the counters. Note that there is hierarchy in the emptying states — Eve cannot move from a state with 0 to a state with +. Let us define the emptying partition of Eve's automaton where for every possible move of Adam there is a cancelling move with additional decrementing of the counters eventually leading to the sink state \top_{00} .

- $\{(\top_{++}, (-4 - e, -1), t) \mid e \in \{0, 1\}, t \in \{\top_{++}, \top_{+0}, \top_{0+}, \top_{00}\}\};$
- $\{(\top_{+0}, (-4 - e, 0), t) \mid e \in \{0, 1\}, t \in \{\top_{+0}, \top_{00}\}\};$
- $\{(\top_{0+}, (-e, -1), t) \mid e \in \{0, 1\}, t \in \{\top_{0+}, \top_{00}\}\};$
- $\{(\top_{00}, (-e, 0), \top_{00}) \mid e \in \{0, 1\}\}.$

Finally, we define transitions connecting the simulating partition of Eve's automaton with the emptying partition. For each state $s_{ab} \in Q'$, Eve has a transition $(s_{ab}, (-1, 0), \top_{ab})$.

Adam is stateless, i.e., he has one state and his moves are self-loops. There are two types of moves: the 0-MOVE, $(0, 0)$, with which Adam agrees that Eve simulated the 2CM correctly and the POSITIVITY CHECK, $(1, 0)$, with which Adam checks whether a flag matches the counter (i.e., Eve simulated incorrectly). Control states of the players are depicted in Figure 3.

To avoid Eve winning trivially every play in the robot game with states, we do not use $(s_{00}, (0, 0))$ as an initial configuration, but instead consider the configuration that is reached in (Q', T') after one step of the run of the machine. We write the configuration after one step as $(s_{\bar{a}\bar{b}}, (y, z))$ and we define $\bar{a} = +, \bar{b} = 0$ if $y = 1$ and $\bar{a} = 0, \bar{b} = +$ if $y = 0$. The initial configuration in the robot game with states is then $(s_{\bar{a}\bar{b}}, (4y, z))$. The effect

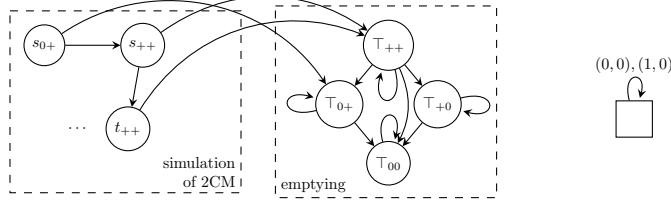


Figure 3: An illustration of state transitions of Eve and Adam.

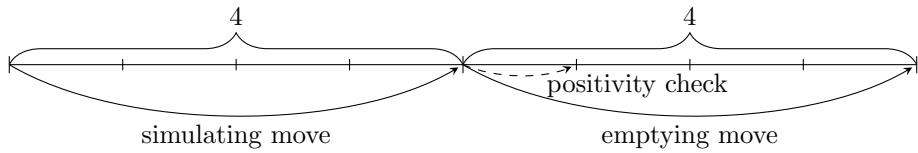


Figure 4: An illustration of changes in an interval when simulating or emptying moves of Eve (solid) or positivity check of Adam (dashed) is applied. Eve's simulating and emptying moves do not change remainder modulo four, while Adam's positivity check changes the remainder by $-1 \pmod{4}$.

of simulating moves, emptying moves and positivity check modulo four is depicted in Figure 4.

Next we prove which player has a winning strategy in the scenarios presented previously.

Lemma 3. *In a sequence where Adam plays only the 0-MOVE and Eve plays only correct SIMULATING MOVES, Adam wins if the 2-counter machine does not reach a configuration with zeros in both counters and Eve wins otherwise.*

PROOF. It is easy to see that correct moves of Eve simulate the 2CM and that a configuration $(s_{00}, (0,0))$ of the 2CM is reachable if and only if the configuration $s_{00}, (0,0)$ is reachable in 2RGS.

Lemma 4. *If Eve plays an incorrect move, i.e., after her turn a flag does not match the counter value (i.e., the flag is + while the counter is 0 or vice versa), then Adam has a winning strategy starting with the POSITIVITY CHECK.*

PROOF. As noted previously, there are two ways Eve can make a mistake regarding the positivity of each counter. If the mistake is in the first counter, then the configuration is $(s_{0b}, (4x, y))$, where $x \geq 1$ or $(s_{+b}, (0, y))$. Alternatively, the mistake is made in the second counter and there are two analogous cases. Namely, either the configuration is $(s_{a0}, (4x, y))$, where $y \geq 1$ or $(s_{a+}, (x, 0))$.

In both subcases Adam plays his POSITIVITY CHECK which changes the parity of the first counter. That is, after Adam's turn, the first counter is $1 \pmod{4}$. It is easy to see that if Eve does not change the parity of the counter back to zero with her following turn, then Adam has a winning strategy. Indeed in this case, he will play his POSITIVITY CHECK if and only if the first counter is $3 \pmod{4}$ and the 0-MOVE otherwise. That is, the configuration is $(s_{ab}, (x, y))$ for some state s_{ab} , $y \geq 1$ and $x = 2, 3 \pmod{4}$ after Adam's turn. Eve can change the parity of the first counter by at most $-1 \pmod{4}$

during her turn for which Adam can compensate and ensure that $x = 2, 3 \pmod{4}$ after his turn. Clearly Eve cannot make the counter 0, as she cannot even make it $0 \pmod{4}$. Thus Eve has to play a move adding $-1 \pmod{4}$ to the first counter as soon as Adam played his POSITIVITY CHECK. The only move for that is $(s_{ab}, (-1, 0), \top_{ab})$ which takes Eve to an emptying state.

Let us consider the mistake in the first counter first. In the first subcase, the emptying state is \top_{0b} and all the transitions from it either do not modify the first counter or subtract one, i.e., Eve cannot reach $(0, 0)$ as Adam can compensate for Eve's moves containing -1 by playing his POSITIVITY CHECK each time. In the second subcase, the emptying state is \top_{+b} where the next transition subtracts 4 from the first counter making it negative and there are no moves that increment the counters. Again, Eve cannot reach $(0, 0)$. The case where Eve makes a mistake with the second counter is proven analogous. In the first subcase, the emptying state is \top_{a0} , the second counter is positive and there are no moves that modify the second counter. That is, Eve cannot reach $(0, 0)$. In the final subcase, the emptying state is \top_{a+} , the next transition subtract 1 from the second counter, making it negative, and there are no moves that increment the counter.

Lemma 5. *Assume that Eve plays only correct SIMULATING MOVES before Adam plays the POSITIVITY CHECK for the first time. If Adam plays the POSITIVITY CHECK, then Eve has a winning strategy starting with an EMPTYING MOVE.*

PROOF. Similarly as in the previous proof, if Eve does not play an EMPTYING MOVE, then Adam has a winning strategy. Now, the configuration is $(s_{ab}, (4x + 1, y))$ after Adam's turn and Eve plays $(s_{ab}, (-1, 0), \top_{ab})$. From that point onward, Eve can empty the counters ensuring that the first counter is $0 \pmod{4}$ and that the flags match the positivity of the counters. That is, every time Adam plays his POSITIVITY CHECK, Eve plays an EMPTYING MOVE subtracting one from the first counter. Eventually, Eve will reach the configuration $(\top_{00}, (0, 0))$ and win the game.

Lemma 6. *If Adam plays only the 0-MOVE and Eve plays an EMPTYING MOVE. Adam has a winning strategy starting with the 0-MOVE.*

PROOF. After Eve's move, the first counter is $3 \pmod{4}$. As in the proof of Lemma 4, Adam ensures that the first counter stays non-zero modulo four and wins the game.

PROOF OF THEOREM 2. Let (A, E) be the robot game with states constructed in this section. Assume first that (Q, T) reaches a configuration in $Q \times \{(0, 0)\}$. Now by Lemma 3, Eve's winning strategy is to respond with the correct SIMULATING MOVES if Adam plays the 0-MOVE, and if Adam plays a POSITIVITY CHECK, then Eve has a sequence of moves described in Lemma 5 that leads to the configuration $(\top_{00}, (0, 0))$.

Assume then that (Q, T) never reaches a configuration in $Q \times \{(0, 0)\}$. We show that Eve does not have a winning strategy. If Adam plays only the 0-MOVE, then, by Lemma 3, Eve does not win by responding with just the correct SIMULATING MOVES. Alternatively, if at some point, she plays either an incorrect SIMULATING MOVE or an EMPTYING MOVE, then by Lemmas 4 and 6, respectively, Adam has winning strategies making sure that a configuration with counter values $(0, 0)$ is not reachable. As we analysed all the possible moves of Eve, we have shown that Eve does not have a winning strategy.

By Theorems 1 and 2, we have the following corollary regarding decidability of two-dimensional robot games with states.

Corollary 7. *Let (A, E) be a robot game with states and \mathbf{x}_0 be the initial vector. It is undecidable whether Eve has a winning strategy to reach $(0, 0)$ from \mathbf{x}_0 . In particular, Adam is stateless and does not modify the second counter.*

4. Stateless robot games in two dimensions

In this section we prove the main result that it is undecidable whether Eve has a winning strategy in a two-dimensional robot game. We prove the claim by constructing a robot game that simulates a robot game with states. In some ways the construction is similar to the construction of a game with states in the previous section as can be seen in similarities of figures 2 and 5. On the other hand, the construction of the stateless game is more complex as the information on two counters, states and state transitions has to be embedded into two-dimensional vectors.

Theorem 8. *Let (A_1, E_1) be a two-dimensional robot game with states where Adam is stateless and does not modify the second counter. There exists a two-dimensional robot game (A, E) where Eve has a winning strategy if and only if Eve has a winning strategy in (A_1, E_1) .*

Similarly to the construction of Section 3, the idea is that in the robot game, Eve and Adam simulate a play of the 2RGS. If one of the players deviates from the play, the opponent has a winning strategy from that point onward. In Figure 5, we present a schematic similar to Figure 2 depicting the possible ways two-dimensional robot games can go. Three of the outcomes have a predetermined winner which does not depend on the 2RGS. In the last case where Eve and Adam correctly simulate the 2RGS, the winner depends on the winner of the 2RGS, i.e., whether Eve has a winning strategy to reach $(s, (0, 0))$, for any state s , or not.

- If Eve’s move corresponds to a move in a play of the 2RGS, that we call a REGULAR MOVE, and Adam replies with his REGULAR MOVE, then iteratively applying only this turn-based interaction, Eve has a winning strategy if and only if she has a winning strategy in the corresponding 2RGS (Lemma 10).
- If Eve’s move incorrectly simulates the 2RGS, then Adam has a winning strategy from this moment on starting with a STATE-CHECK that makes Eve’s target unreachable (Lemma 11).
- On the other hand, if Adam plays his STATE-CHECK following a correct REGULAR MOVE of Eve, then Eve has a winning strategy from this moment on starting with a STATE-DEFENCE MOVE allowing Eve to empty both counters and reach $(0, 0)$ (Lemma 12).
- Finally, if Eve plays a STATE-DEFENCE MOVE instead of a REGULAR MOVE, in that case Adam has a winning strategy starting by playing his REGULAR MOVE (Lemma 13).

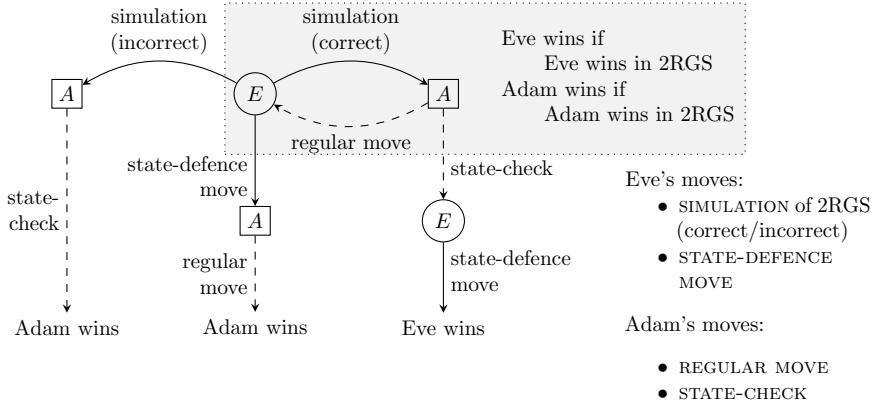


Figure 5: Progress of 2RG. Here solid arrows are moves of Eve and dashed are moves of Adam.

Intuitively, we encode the states as powers of 8 such that the coefficient of 8^i is 1 if and only if Eve's state in robot games with states is s_i . When the state changes from s_i to s_j , $-8^i + 8^j$ is added to the second counter. We represent states as coefficients of powers of eight because we need the extra space smaller bases do not possess.

It is easy to see that simply encoding states as powers of 8 is not enough as incorrect transitions can result in a correct configuration. For example, if the configuration of the 2RGS is $(s_i, (x, y))$ and moves corresponding to $(s_j, (a, b), s_k)$ and $(s_k, (c, d), s_j)$ are used, the resulting configuration corresponds to $(s_i, (x + a + c, y + b + d))$. Another way to cheat is to use carries as incrementing the coefficient of 8^i eight times is indistinguishable from incrementing the coefficient of 8^{i+1} once. Both types of cheating can be countered with Adam's STATE-CHECKS.

We now show how we embed the states and state transitions into the second counter of the game. Similarly to how in the previous section we created additional space in the first counter by multiplying the moves modifying the first counter by four, we multiply the second counter by $4 \cdot 8^n$, where $n = m + 7$ and m is the number of states, creating enough space to store all the needed information of the underlying automaton. The multiplication by $4 \cdot 8^n$ rather than just 8^n has two purposes. The first one is similar to multiplying the first counter by four in the Section 3. Namely, certain moves will move between different intervals modulo $4 \cdot 8^n$ ensuring the correct response from the opponent. This is illustrated in Figure 6. The second purpose is to ensure that above described cheating with carries is not possible. A configuration in $Q \times \mathbb{Z}^2$ is mapped to a vector in \mathbb{Z}^2 by $(s_i, (c_1, c_2)) \mapsto (c_1, c_2 \cdot 4 \cdot 8^n + 8^i)$.

Before presenting the detailed constructions of Eve's and Adam's moves, we note that we can assume that the 2RGS has the information on the positivity of the counters and players have to update the information correctly. Indeed, this was done in the previous section by using flags 0 and +. Recall that because of this, the first counter is incremented and decremented by 4. By this assumption, we can denote the states of Eve by s_{ab} as before. We also assume that Eve's automaton is without self-loops as they would allow Eve to modify the counters without modifying coefficients of the states. Let Q be the set of states of Eve in 2RGS. We create an emptying gadget for Eve similar to the one

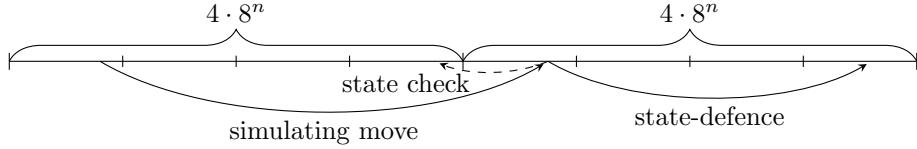


Figure 6: An illustration of changes in interval when simulating or state-defence moves of Eve (solid) or state check of Adam (dashed) is applied.

constructed in the previous reduction. To avoid self-loops, there are seven emptying states, $\{\top_{ab}, \top'_{ab} \mid a, b \in \{0, +\}\} \setminus \{\top'_{00}\}$. The state \top'_{00} is not needed as \top_{00} will not have any moves from it. The moves in the emptying gadget are as in the emptying gadget constructed in Section 3 but instead of self-loops, the transitions are between primed and unprimed versions of the states.

- $\{(\top_{++}, (-4, -1) - \alpha, t) \mid \alpha \in A_1, t \in \{\top'_{++}, \top_{+0}, \top'_{+0}, \top_{0+}, \top'_{0+}, \top_{00}\}\};$
 $\{(\top'_{++}, (-4, -1) - \alpha, t) \mid \alpha \in A_1, t \in \{\top'_{++}, \top_{+0}, \top'_{+0}, \top_{0+}, \top'_{0+}, \top_{00}\}\};$
- $\{(\top_{+0}, (-4, 0) - \alpha, t) \mid \alpha \in A_1, t \in \{\top'_{+0}, \top_{00}\}\};$
 $\{(\top'_{+0}, (-4, 0) - \alpha, t) \mid \alpha \in A_1, t \in \{\top_{+0}, \top_{00}\}\};$
- $\{(\top_{0+}, (0, -1) - \alpha, t) \mid \alpha \in A_1, t \in \{\top'_{0+}, \top_{00}\}\};$
 $\{(\top'_{0+}, (0, -1) - \alpha, t) \mid \alpha \in A_1, t \in \{\top_{0+}, \top_{00}\}\};$

We denote $\mathcal{T} = \{\top_{++}, \top'_{++}, \top_{0+}, \top'_{0+}, \top_{+0}, \top'_{+0}, \top_{00}\}$. We think of elements of $Q \cup \mathcal{T} \cup \{\top_{00}\}$ as integers in $\{0, \dots, n-1\}$ such that $\top_{00} = 0, \top'_{0+} = n-6, \top_{0+} = n-5, \top'_{+0} = n-4, \top_{+0} = n-3, \top'_{++} = n-2, \top_{++} = n-1$. We give names for update vectors that we often use:

$$\begin{aligned} \text{ADD}(1, x) &:= (x, 0); & \text{MOVE}(j, k) &:= (0, -8^j + 8^k), \text{ for } 0 \leq j, k \leq n-1; \\ \text{ADD}(2, x) &:= (0, 4x \cdot 8^n); & \text{CHECK}(i) &:= (0, -5 \cdot 8^i - 8^n), \text{ for } n-6 \leq i \leq n-1. \end{aligned}$$

The initial vector of the robot game is $\text{ADD}(1, x) + \text{ADD}(2, y) + \text{MOVE}(\top_{00}, s)$, that is, $(x, 4y \cdot 8^n + 8^s - 8^0)$, where $(s, (x, y))$ is the initial configuration in the robot game with states. In the next example we illustrate how the update vectors modify the counters.

Example 9. Let (A_1, E_1) be a two-dimensional robot game with states where Eve has two states, $s = 1$ and $t = 2$, and the initial configuration $(s, (1, 0))$. Next we present a set of configurations in 2RG obtained from the corresponding initial configuration when we

apply ADD(1, -1), ADD(2, 1), MOVE(s, t), CHECK(i) in succession:

$$\begin{aligned}
& \overbrace{(1, 0 \cdot 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0)}^{2RGS \text{ counters}} \\
& \quad \downarrow \text{ADD}(1, -1) \\
& (0, 0 \cdot 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
& \quad \downarrow \text{ADD}(2, 1) \\
& (0, 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \\
& \quad \downarrow \text{MOVE}(s, t) \\
& (0, 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0) \\
& \quad \downarrow \text{CHECK}(8) \\
& (0, 3 \cdot 8^9 - 5 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0).
\end{aligned}$$

Now we present the moves of the players. Adam has two types of moves: REGULAR MOVES that correspond to the moves in the 2RGS and STATE-CHECK MOVES, $\{\text{CHECK}(i) \mid i \in \mathcal{T}\}$. The moves of Eve correspond to moves in E_1 where incrementing and decrementing of the second counter is by $4 \cdot 8^n$ rather than by 1. Let $(s, (x, y), t) \in E_1$, then $\text{ADD}(1, x) + \text{ADD}(2, y) + \text{MOVE}(s, t) = (x, 4y \cdot 8^n - 8^s + 8^t) \in E$. We call these moves REGULAR MOVES. We also need a move for Eve to finish the simulation by removing any values corresponding to the automaton if the state is s_{00} . That is, we add moves $\{\text{MOVE}(s_{00}, \top_{00}) - \alpha \mid \alpha \in A_1\}$. The other type of moves, STATE-DEFENCE MOVES, are used to empty the counters. As in the previous construction, Eve will be able to cancel every Adam's move and decrement the counters at the same time.

Finally, we define moves connecting the simulating partition of Eve's automaton with the emptying partition. For each state $s_{ab} \in Q$ where a, b are not both zero, Eve has a move $\{\text{MOVE}(s_{ab}, k) - \text{CHECK}(i) \mid (a, b) \in \{0, +\}^2 \setminus \{(0, 0)\}, k \in \{\top_{ab}, \top'_{ab}\}, k \neq i, i \in \mathcal{T}\}$. For s_{00} , Eve has a move $\{\text{MOVE}(s_{00}, \top_{00}) - \text{CHECK}(i) \mid i \in \mathcal{T}\}$.

Adam's move	Eve's move
$\alpha \in A_1$	$\{\text{ADD}(1, -4) + \text{ADD}(2, -1) - \text{MOVE}(j, k) - \alpha \mid j, k \in \{\top_{++}, \top'_{++}\}, j \neq k\}$
	$\{\text{ADD}(1, -4) - \text{MOVE}(j, k) - \alpha \mid j, k \in \{\top_{+0}, \top'_{+0}\}, j \neq k\}$
	$\{\text{ADD}(2, -1) - \text{MOVE}(j, k) - \alpha \mid j, k \in \{\top_{0+}, \top'_{0+}\}, j \neq k\}$
	$\{\text{ADD}(1, -4) + \text{ADD}(2, -1) + \text{MOVE}(j, k) - \alpha \mid j \in \{\top_{++}, \top'_{++}\}, k \in \mathcal{T}, j \neq k\}$
	$\{\text{ADD}(1, -4) + \text{ADD}(2, -1) + \text{MOVE}(j, 1) - \alpha \mid j \in \{\top_{++}, \top'_{++}\}\}$
	$\{\text{ADD}(1, -4) + \text{MOVE}(j, 1) - \alpha \mid j \in \{\top_{+0}, \top'_{+0}\}\}$
CHECK(i)	$\{\text{ADD}((1, -4e_1) + \text{ADD}(2, -e_2) - \text{CHECK}(i)) \mid e_1, e_2 \in \{0, 1\}\}$
	$\{\text{ADD}(1, -4) + \text{ADD}(2, 1) + \text{MOVE}(j, k) - \text{CHECK}(i) \mid i, j \neq k,$ $j \in \{\top_{++}, \top'_{++}\}, k \in \mathcal{T}\}$
	$\{\text{ADD}(1, -4) + \text{ADD}(2, -1) + \text{MOVE}(j, 1) - \text{CHECK}(i) \mid j \in \{\top_{++}, \top'_{++}\}\}$
	$\{\text{ADD}(1, -4) + \text{MOVE}(j, 1) - \text{CHECK}(i) \mid j \in \{\top_{+0}, \top'_{+0}\}\}$
	$\{\text{ADD}(2, -1) + \text{MOVE}(j, 1) - \text{CHECK}(i) \mid j \in \{\top_{0+}, \top'_{0+}\}\}$

Next we prove which player has a winning strategy in the scenarios presented previously.

Lemma 10. *If both players only play REGULAR MOVES and Eve plays only correct REGULAR MOVES, then Eve has a winning strategy if and only if she has a winning strategy in two-dimensional robot games with states.*

PROOF. It is easy to see that REGULAR MOVES of the players simulate the 2RGS and that Eve has a winning strategy to reach a configuration $(s_{00}, (0, 0))$ of the 2RGS if and only if she has a winning strategy to reach the vector $(0, 0 \cdot 4 \cdot 8^n + 8^{s_{00}} - 8^{\top_{00}})$ in 2RG after which Eve wins by playing $\text{MOVE}(s_{00}, \top_{00}) - \alpha$, where α is the REGULAR MOVE played by Adam.

Lemma 11. *If Eve plays an incorrect move, i.e., after her turn the coefficient of some 8^s is -1 or the coefficient of $8^{\top_{00}}$ is zero, then Adam has a winning strategy starting with a STATE-CHECK.*

PROOF. First, we prove that Eve loses if a coefficient corresponding to a state of 2RGS is negative after one of her turns. A coefficient corresponding to a state of 2RGS can only be increased, namely incremented, by Eve's REGULAR MOVES. Hence, if one of the coefficients becomes negative, then Adam wins by playing a STATE-CHECK move. The reasoning is now similar to the usage of the POSITIVITY CHECK in Lemma 4. We consider the second counter modulo $4 \cdot 8^n$. Before Adam's STATE-CHECK, the configuration is in $[0, 8^n) \pmod{4 \cdot 8^n}$ and after the check in $[3 \cdot 8^n, 4 \cdot 8^n) \pmod{4 \cdot 8^n}$. If Eve does not play a STATE-DEFENCE MOVE (a move containing a $\text{CHECK}(i)$), then Adam has a winning strategy by playing a STATE-CHECK if the second counter is not in $[3 \cdot 8^n, 4 \cdot 8^n) \pmod{4 \cdot 8^n}$ and a REGULAR MOVE otherwise (recall that Adam's REGULAR MOVES do not modify the second counter). Thus Eve has to play a STATE-DEFENCE MOVE which does not make the negative coefficient non-negative. Now at least one of the coefficients in \mathcal{T} is non-zero, say i . Adam will play $\text{CHECK}(i)$ forcing Eve to play a move containing $-\text{CHECK}(i)$ which will make another coefficient in \mathcal{T} non-zero. As long as Adam keeps playing the correct STATE-CHECK, Eve cannot make all the coefficients zero and thus cannot win.

The second case where a coefficient of some state in \mathcal{T} is negative has been proven above. For the final case, where the coefficient of $8^{\top_{00}}$ is zero, we consider the next move of Eve. During her next turn, Eve has to play a move containing $\text{MOVE}(s, t)$ making the coefficient of 8^s negative, which has been covered previously.

Lemma 12. *Assume that Eve plays only correct REGULAR MOVES before Adam plays a STATE-CHECK for the first time. If Adam plays a STATE-CHECK, then Eve has a winning strategy starting with a STATE-DEFENCE MOVE.*

PROOF. Similarly as in the previous proof, if Eve does not play a STATE-DEFENCE MOVE, then Adam has a winning strategy. Now, Eve plays the STATE-DEFENCE MOVE $\text{MOVE}(s_{ab}, k) - \text{CHECK}(i)$ where s_{ab} is the non-zero coefficient, $\text{CHECK}(i)$ is the STATE-DEFENCE MOVE Adam played and $k \in \{\top_{ab}, \top'_{ab}\}$, $k \neq i$. From that point onward, Eve can empty the counters ensuring as she has emptying moves with an opposite move of Adam. Eventually, Eve will reach the configuration $(0, 0)$ and win the game.

Lemma 13. *If Adam plays only REGULAR MOVES and Eve plays a STATE-DEFENCE MOVE, then Adam has a winning strategy starting with a REGULAR MOVE.*

PROOF. Since all STATE-DEFENCE MOVES subtract -8^n from the second counter, after Eve's move, the counter is in $[8^n, 2 \cdot 8^n] \pmod{4 \cdot 8^n}$. As in the proof of Lemma 11, Adam ensures that the second counter does not return to the interval $[0, 8^n] \pmod{4 \cdot 8^n}$.

PROOF (PROOF OF THEOREM 8). Let (A, E) be the robot game constructed in this section. Assume first that Eve has a winning strategy in (A_1, E_1) . Now, Eve's winning strategy in two-dimensional robot games is to follow the strategy of (A_1, E_1) as long as Adam plays REGULAR MOVES which is a winning strategy by Lemma 10. If Adam plays a STATE-CHECK, then Eve responds according to the winning strategy of Lemma 12.

Assume then that Adam has a winning strategy in (A_1, E_1) and Eve has a winning strategy in (A, E) . If Adam plays only REGULAR MOVES, then by Lemma 10, Eve does not win by playing just the correct the correct SIMULATING MOVES. That is, Eve has to, at some point, either play an incorrect SIMULATION MOVE or play a STATE-DEFENCE MOVE. By Lemmas 11 and 13, Adam has winning strategies for both cases. As we analysed all the possible moves of Eve, we have shown that Eve does not have a winning strategy.

Corollary 14. *Let (A, E) be a two-dimensional robot game and an initial vector \mathbf{x}_0 . It is undecidable whether Eve has a winning strategy to reach $(0, 0)$ from \mathbf{x}_0 .*

Corollary 14 follows from Corollary 7 and Theorem 8.

5. Robot games with states in dimension one

In this section we consider games in dimension one. First, we recall some known results.

Theorem 15 ([28]). *Deciding which player wins in a one-dimensional counter reachability game is EXPSPACE-complete.*

Theorem 16 ([20]). *Deciding which player wins in a one-dimensional robot game is EXPTIME-complete.*

As robot games are a special case of robot games with states, we can inherit the lower bound. That is, the 1RGS are EXPTIME-hard. On the other hand it is easy to construct a counter reachability game out of a robot game with states by storing information on the state of Eve in the 1RGS in the states of Adam and vice versa. That is, robot games with states are in EXPSPACE.

Lemma 17. *Deciding which player wins in a one-dimensional robot game with states is EXPSPACE.*

PROOF. Let (A, E) be a 1RGS and $z_0 \in \mathbb{Z}$ the initial integer. We construct a counter reachability game (V, F) where Eve has a winning strategy if and only if Eve has a winning strategy in (A, E) . Eve's states are $V_E = \{s_t \mid s \in Q_E, t \in Q_A\}$ and Adam's states are $V_A = \{t_s \mid t \in Q_A, s \in Q_E\}$. The edges of the graph are $F = \{(s_t, z, t_{s'}) \mid (s, z, s') \in E\} \cup \{(t_s, z, s_{t'}) \mid (t, z, t') \in A\}$. It is clear that Eve has a winning strategy in (A, E) from z_0 if and only if Eve has a winning strategy in (V, F) from z_0 . As deciding the winner in the 1CRG is EXPSPACE-complete, also deciding the winner in the 1RGS is EXPSPACE. \square

We provide the matching tight lower bound, showing that one-dimensional robot games with states are EXPSPACE-complete. That is, we show that the 1RGS are EXPSPACE-hard. To prove this, we show how, for any one-dimensional counter reachability game, to construct a one-dimensional robot game with states such that the same player wins in both games. The idea is for Eve to have the whole graph, including Adam's states, as her states and Adam have no states. Adam has three moves, two to tell Eve which edge to pick if the state was initially Adam's, and one to do nothing if that's not the case.

Theorem 18. *One-dimensional robot games with states are EXPSPACE-complete.*

First we consider a simple modification to a 1CRG. We can assume that in every Adam's state there are at most two outgoing edges. Indeed, let t be Adam's vertex with k outgoing edges, we replace it by a chain t_1, \dots, t_{k-1} such that i th edge (t, z, t') is (t_i, z, t') . Finally, we connect the vertices with edges $(t_i, 0, t_{i+1})$ for $i \in \{1, \dots, k-1\}$ and $(t, 0, t_1)$.

Next, we show the gadgets for different moves in the 1CRG. At this state, for simplicity, we assume that both players will play in good faith and will simulate the 1CRG correctly. Later on, we'll construct an additional gadget for Eve and show that if one of the player cheats, then the other can catch the cheating and has a winning strategy.

Now, there are three types of transitions: from Eve's state or from Adam's state which has either one or two outgoing transitions. We construct gadgets for each case. Let's first consider the cases where Adam does not make a decision. That is moves (s, z, r) and (t, z, r) , where $z \in \mathbb{Z}$, $s \in V_E$, $r \in V$, $t \in V_A$ and $\deg(t) = 1$. In robot game with states, Eve has moves $(s, 4z, r)$ and $(t, 4z, r)$, where $s, r, t \in Q_E$, respectively, and Adam has a move $(\top, 0, \top)$. The moves are depicted in Figure 7.

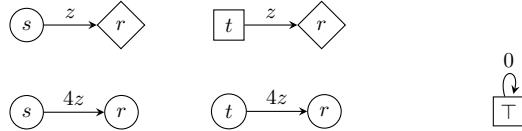


Figure 7: Moves in a 1CRG (top) and the corresponding moves in the 1RGS (bottom).

The final case where Adam has to make a choice is slightly more complicated. As Eve is simulating the whole graph of the 1CRG, Adam needs to indicate to her which edge he would have picked. In the 1CRG the moves are $(t, y, p), (t, x, q)$, where $p, q \in V$ and $t \in V_A$ and $\deg(t) = 2$. In robot game with states, Eve has a gadget with moves $(t, 4y - 1, p), (t, 4x + 1, q)$, and Adam has moves $(\top, 1, \top)$ and $(\top, -1, \top)$. The moves are depicted in Figure 8. By multiplying all the old labels by 4, we have created extra space to store the information about which edge Eve is supposed to pick.



Figure 8: Moves in a 1CRG (left) and the corresponding moves in the 1RGS (right).

Finally, we need to make sure that Adam does not abuse his moves, i.e., does not indicate his choice when he should not. For this, we create a gadget similar to Adam's

state transition, which Eve can enter and add ± 4 emptying the counter while at the same time cancelling whatever Adam plays. To do so, we design an emptying gadget of Eve consisting of one state \perp . The moves are $(\perp, \pm 4 + 1, \perp)$, $(\perp, \pm 4 - 1, \perp)$ and $(\perp, \pm 4, \perp)$. The emptying gadget is connected to states of Eve with moves $(s, \pm 1, \perp)$ for every state $s \in V_E$ or $s \in V_A$ and $\deg(s) = 1$, and with $(t, 0, \perp)$, where $t \in V_A$ and $\deg(t) = 2$. The control states of the players are depicted in Figure 9.

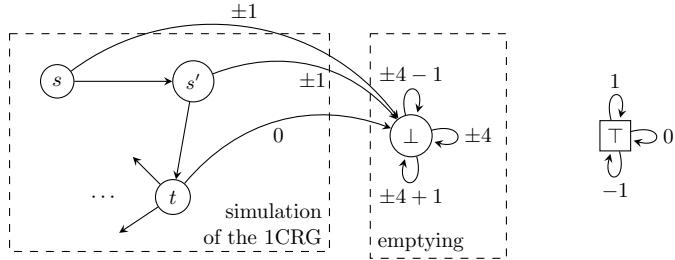


Figure 9: An illustration of state transitions of Eve and Adam.

Next we consider all possible plays of Adam and Eve, and show that if the player plays incorrectly, the opponent has a winning strategy. The possible ways the game can progress are listed in Figure 10. First, we informally describe the incorrect moves and how the opponent can deal with them.

Adam can play incorrectly by either playing ± 1 even though Eve is not in a state where Adam has to make a decision, or by playing 0 if Eve is. In the first case, Eve can play the opposite move and move to \perp , after which she can counter any move Adam plays whilst emptying the counter. In the latter case, Eve moves to \perp without modifying the counter and again she can empty the counter while nullifying the moves of Adam.

Eve can play incorrectly by either moving to the emptying gadgets before Adam made an incorrect move or by not making the correct decision according to what Adam has played, that is, playing $4y - 1$ after Adam played -1 or $4x + 1$ after Adam played 1. In the both cases, Adam can ensure that the counter will never be $0 \pmod{4}$.

First we prove two lemmas regarding incorrect moves by Adam and prove that Eve has winning strategies.

Lemma 19. *Let the configuration be $[s, \dot{T}, 4z]$, where $z \in \mathbb{Z}$ and $s \in V_E$ or $s \in V_A$ and $\deg(s) = 1$. If Adam plays $(\dot{T}, 1, \dot{T})$, then Eve has a winning strategy starting with $(s, -1, \perp)$. Similarly, if Adam plays $(\dot{T}, -1, \dot{T})$, then Eve has a winning strategy starting with $(s, 1, \perp)$.*

PROOF. After Adam's move, the configuration is $[\dot{s}, \dot{T}, 4z + 1]$ and after Eve's move, the configuration is $[\perp, \dot{T}, 4z]$. After this, Eve can cancel Adam's move while emptying the counter at the same time. In the case Adam played $(\dot{T}, -1, \dot{T})$, then Eve's winning strategy is the same after she played $(s, 1, \perp)$. \square

Lemma 20. *Let the configuration be $[t, \dot{T}, 4z]$, where $z \in \mathbb{Z}$ and $t \in V_A$ and $\deg(t) = 2$. If Adam plays $(\dot{T}, 0, \dot{T})$ then Eve has a winning strategy starting with $(t, 0, \perp)$.*

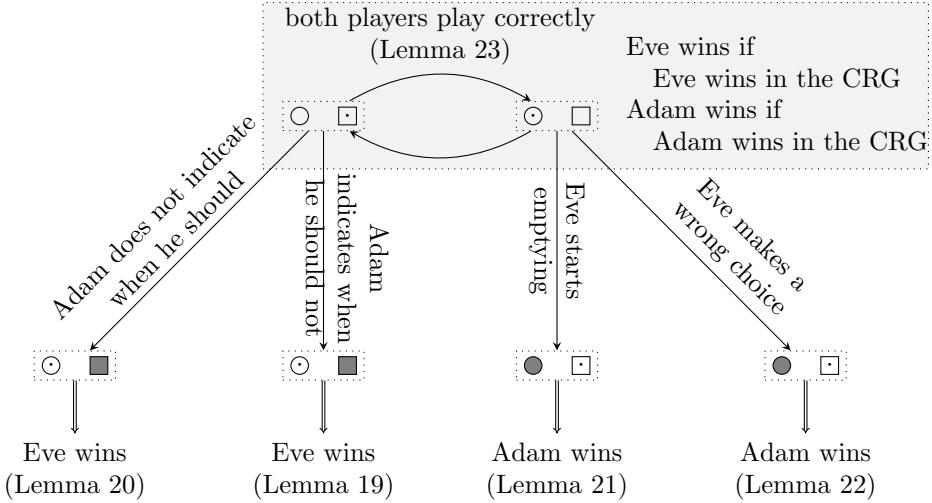


Figure 10: Progress of a one-dimensional robot game with states.

PROOF. After Adam's move, the configuration is $[\dot{t}, \top, 4z]$ and after Eve's move, the configuration is $(\perp, \dot{\top}, 4z)$. As in previous lemma, Eve can empty the counter while cancelling Adam's move. \square

Next we prove a lemma, where Eve moves to her emptying gadget and prove that Adam has a winning strategy.

Lemma 21. *Let the configuration be $[\dot{s}, \top, 4z]$ where $z \in \mathbb{Z}$ and $s \in V_E$ or $s \in V_A$ and $\deg(s) = 1$. If Eve moves to \perp with a move $(s, 1, \perp)$ or a move $(s, -1, \perp)$, then Adam has a winning strategy starting with $(\top, 1, \top)$ or $(\top, -1, \top)$ respectively.*

PROOF. After Eve's move, the configuration is $[\perp, \dot{\top}, 4z \pm 1]$ and after Adam's move, the configuration is $[\perp, \dot{\top}, 4z \pm 2]$. From this moment onward, Adam can ensure that the counter is not $0 \pmod{4}$. Thus, Eve cannot reach counter value 0 and cannot win. \square

Finally, we consider the case where Adam tells Eve his non-deterministic choice with 1 or -1 , and Eve responds incorrectly by playing a move with 1 or -1 , respectively.

Lemma 22. *Let the configuration be $[\dot{t}, \top, 4z+1]$, where $z \in \mathbb{Z}$ and $t \in V_A$ and $\deg(t) = 2$. If Eve plays the move $(t, 4y+1, p)$, then Adam has a winning strategy starting with $(\top, 0, \top)$. Symmetrically, if the configuration is $[\dot{t}, \top, 4z-1]$ and Eve plays the move $(t, 4x-1, q)$, then Adam has a winning strategy.*

PROOF. After Eve's move, the configuration is $[p, \dot{\top}, 4(z+y)+2]$ and Adam with his moves can ensure that the counter is not $0 \pmod{4}$. That is, Eve cannot reach counter value 0 and cannot win. Symmetrically, if after Eve's move the configuration is $[p, \dot{\top}, 4(z+x)-2]$, then Adam can ensure that the counter is not $0 \pmod{4}$ and Eve cannot win. \square

Next we prove that if both players play correctly, the winner is the same as in the one-dimensional counter reachability game.

Lemma 23. *If in the one-dimensional robot game with states constructed previously Eve plays*

- the move $(t, 1, p)$ if the configuration is $[\dot{t}, \top, 4z - 1]$ for some $z \in \mathbb{Z}$ and $t \in V_E$ and $\deg(t) = 2$,
- the move $(t, -1, p)$ if the configuration is $[\dot{t}, \top, 4z + 1]$ for some $z \in \mathbb{Z}$ and $t \in V_E$ and $\deg(t) = 2$

and never moves to \perp , and Adam plays

- the move $(\top, 0, \top)$ if the configuration is $[s, \dot{\top}, 4z]$, for some $z \in \mathbb{Z}$ and $s \in V_E$ or $s \in V_A$ and $\deg(s) = 1$,
- a move $(\top, -1, \top)$ or $(\top, 1, \top)$ if the configuration is $[\dot{t}, \top, 4z]$, for some $z \in \mathbb{Z}$ and $t \in V_A$ and $\deg(t) = 2$,

then Eve has a winning strategy if and only if she has a winning strategy in the one-dimensional counter reachability game.

PROOF. It is easy to see that these moves simulate the counter reachability game and that Eve has a winning strategy to reach the configuration $[f, 0]$ of the 1CRG if and only if she has a winning strategy to reach the configuration $[\dot{f}, \top, 0]$ in the 1RGS. \square

We are ready to prove the main theorem.

Theorem 3. *The one-dimensional robot games with states are EXPSPACE-complete.*

PROOF. By Lemma 17, deciding the winner is in EXPSPACE. It remains to be proven that it is also EXPSPACE-hard. Let (V, F) be a 1CRG with an initial counter z_0 . Let (A, E) be the robot game with states constructed from (V, F) . Assume first that Eve has a winning strategy in (V, F) . Now, Eve's winning strategy in the one-dimensional robot game (A, E) is to play according to the winning strategy of (V, F) if the configuration $[\dot{s}, \top, 4z]$ where $s \in V_E$ or $s \in V_A$ and $\deg(s) = 1$. If the configuration is $[\dot{t}, \top, 4z - 1]$ or $[\dot{t}, \top, 4z + 1]$ where $t \in V_A$, $\deg(t) = 2$, then moves $(t, 4x + 1, q)$ or $(t, 4y - 1, p)$, respectively. This is a winning strategy by Lemma 23. If the configuration is $[\dot{s}, \top, 4z \pm 1]$ where $s \in V_E$ or $s \in V_A$ and $\deg(s) = 1$, then Eve has a winning strategy by Lemma 19. If the configuration is $[\dot{t}, \top, 4z]$ where $t \in V_A$ and $\deg(t) = 2$, then Eve has a winning strategy by Lemma 20.

Assume then that Adam has a winning strategy in (V, F) and Eve has a winning strategy in (A, E) . By Lemma 23, Adam has a winning strategy if the players simulate the 1CRG correctly. That is, Eve has to, at some point, either move to an emptying gadget, or play $(t, 4x \pm 1, s)$ when the configuration is $[\dot{t}, \top, 4z \mp 1]$. By Lemmas 21 and 22, Adam has winning strategies for both cases. As we have analysed all the possible moves of Eve, we have shown that Eve does not have a winning strategy. \square

6. Flat robot games with states

There is an interesting complexity difference between games of Theorem 16 and Theorem 3. When the (stateless) robot game is extended by allowing Eve to have an

internal state structure and keeping Adam stateless, the complexity of deciding the winner increases from EXPTIME to EXPSPACE. In this section we study a natural dual question — does keeping Eve stateless and allowing Adam to have an internal structure result in a similar increase? We study this question by considering robot games where Eve is stateless and Adam's states are flat (i.e., the underlying graph is directed acyclic graph with self-loops), called flat robot games with states (FRGS). The state structure of a FRGS is depicted in Figure 11. The main result of the section is that deciding the winner in the FRGS is in EXPTIME. Note, that as the stateless robot games are also flat robot games with states, we have inherited EXPTIME-hard lower-bound.

Remark 1. Let (A, E) be a robot game. If the winning set is non-trivial then it is either $d\mathbb{Z}$, for some integer d , or $R \cup U \subseteq \mathbb{Z}^+$, where $U = \{x \in d\mathbb{Z} \mid x > b\}$ and R consists of the winning values on the finite arena $[0, b]$, or $R' \cup U' \subseteq \mathbb{Z}^-$, where $U' = \{x \in d\mathbb{Z} \mid x < b\}$ and R' consists of the winning values on the finite arena $[b, 0]$.

We say that the *size* of a robot game (A, E) is $\sum_{x \in A \cup E} \log(|x|)$. We also define the size of winning sets in a natural way. If the winning set is of the form $d\mathbb{Z}$, then the size is $\log(|d|)$. If the winning set is of the form $R \cup U$, where as above, R is a finite set and $U = \{x \in d\mathbb{Z} \mid x > b\}$ or $U = \{x \in d\mathbb{Z} \mid x < b\}$, then the size is $\sum_{x \in R} \log(|x|) + \log(|d|)$. Note that the size of a winning set is exponential in the size of the game.

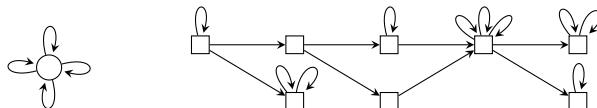


Figure 11: An example of a flat robot game with states (FRGS).

Before considering arbitrary flat graphs of Adam, we consider a simpler case where there are three types Adam's moves: self-loops in state t_0 , self-loops in state t_1 and transitions from t_0 to t_1 .

The idea is that there are two stateless robot games when moves are restricted to self-loops and additional moves connecting the games. The algorithm of [20] not only computes whether the given initial value z_0 is winning for Eve, but it computes the set of all winning values. We can use the algorithm to compute winning sets for both games and then connect the two games using the transitions between t_0 and t_1 .

Example 4. Consider a one-dimensional flat robot game with states where Eve's moves are $\{(s, -3, s), (s, -6, s), (s, -7, s), (s, -8, s)\}$ and Adam's moves are

$$\{(t_0, -3, t_0), (t_0, -6, t_0), (t_0, 0, t_1), (t_1, -7, t_1), (t_1, -8, t_1)\}.$$

It is easy to compute the winning sets for games restricting to t_0 and t_1 : $W_0 = 9\mathbb{Z}^+$ and $W_1 = \{0, 14, 15, 25, 28, 29, 30, 39, 40, 41, 42, 43, 44, 45\} \cup \{x \mid x \geq 50\}$, respectively.

We notice that, for example, 9 is not a winning value in the flat robot game with states. Indeed, while by staying in t_0 , Adam loses, if he instead moves to t_1 , then after Eve's turn the counter will be 1, 2, 3 or 6. None of which is a winning value when restricting to t_1 . On the other hand, all the other winning values, that is $9k$ where $k > 1$, can reach 0 only by reaching 9 first. That is, Eve does not have any winning values. This is illustrated in Figure 12.

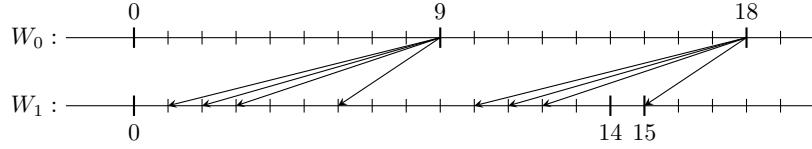


Figure 12: An illustration of connecting winning sets in a flat robot game with states.

For this special case, there are three steps needed to compute the winning set of the game.

- Compute the winning sets of restricted games, W_0 and W_1 .
- Compute the forbidden values F in W_0 , that is, all the values in W_0 from which there exists a move (t_0, z, t_1) of Adam such that for any move (s, x, s) of Eve, the resulting value is not in W_1 .
- Finally, check whether values of F are avoidable in W_1 . That is, whether there exists a winning strategy from the initial value z_0 to 0 that does not visit any values of F .

The first step can be done in EXPTIME using the algorithm for robot games. The second and third steps require some additional considerations as the sets are potentially infinite. In the game of the previous example, if the initial value is z_0 , then it is not important to check which forbidden values larger than z_0 are avoidable and which are not. On the other hand, it is easy to see that in general case, it is not a simple matter of discarding larger values than z_0 (assuming that z_0 is positive). By Remark 1, the winning set of a robot game has a particular structure. In a similar manner, the set of forbidden values constructed from two winning sets have some structure which allows us to compute whether the values are avoidable. Now, there are two sets of forbidden values, one resulting from the finite set of R , $F_{\text{fin}} = \{f_1, \dots, f_k\}$, and a regular but infinite set of values resulting from U , F_{inf} . Even though, F_{inf} is infinite, it is semi-linear (and in fact linear when Adam has two states). We can extract a finite set of forbidden values, F' , such that $F_{\text{inf}} = \bigcup_{i=0}^{\infty} F' + i\ell$ for some $\ell \in \mathbb{Z}$. Now, we have two finite sets of forbidden values for which it is easy to check whether the values are avoidable. We can use the attractor construction found in Chapter 2 of [29] which solves the game in polynomial time. In our example, $F = \{9, 27\}$ and 9 is reachable only from one winning value, namely 18. On the other hand, 9 is the only winning value reachable from 18, so 9 is not avoidable.

Lemma 5. *Let (A_0, E) and (A_1, E) be two robot games and T a set of Adam's moves connecting the two games. Let W_0 and W_1 be their respective winning sets. The set $F = \{x \in W_0 \mid \exists z \in T \forall (s, y, s) \in E : x + z + y \notin W_1\}$ can be computed in polynomial time in sizes of W_0 and W_1 .*

PROOF. There are several cases to consider. First we have two trivial cases when one of the winning sets is trivial, i.e., $\{0\}$. If the winning set W_0 is trivial, then $F = \{0\}$. If the winning set W_1 is trivial, then $F = W_0$. Another obvious case is when the winning set $W_0 \subseteq \mathbb{Z}^+$ and $W_1 \subseteq \mathbb{Z}^-$ (or the symmetric situation), then there are only finitely many points in W_0 from which it is possible to reach W_1 . Thus $F = W_0 \setminus X$ where X is a finite subset of $(0, a]$ for some a bounded by $\min(E) + \min(T)$. There remains four cases.

1. $W_0 = d\mathbb{Z}$ and $W_1 = d'\mathbb{Z}$, or
2. $W_0 = R \cup U$ and $W_1 = d'\mathbb{Z}$, or
3. $W_0 = R \cup U$ and $W_1 = R' \cup U'$, or
4. $W_0 = d\mathbb{Z}$ and $W_1 = R' \cup U'$.

Recall that R and R' are finite and $U = \{x \in d\mathbb{Z} \mid x > b\}$. Consider the first case. Let $\ell = \text{lcm}(d, d')$. We can partition the integer line into intervals of length ℓ and effectively compute all the forbidden values F' in the interval. Clearly, the forbidden values in one interval, will be also forbidden in the other intervals. The set of all forbidden values is $F = \{f + \ell i \mid f \in F', i \in \mathbb{Z}\}$.

The next case can be divided into two parts, first finding forbidden values in R and then in U . Finding the forbidden values in R is easy as there are only finitely many possible values. Finding the forbidden values in U can be done as for the first case. The third and fourth cases are done similarly but now we also have to take the finite set R' into account. In all three cases, the set of forbidden values is $F = \{f_1, \dots, f_k\} \cup \left[\bigcup_{i=0}^{\infty} F' + i\ell \right]$, where $|F'| < \infty$ and $f > f_j$ for all indexes j and $f \in F'$. \square

Lemma 6. *Let (A_0, E) be a robot game and $W_0 \subseteq \mathbb{Z}^+$ its winning set. Let $F_{\text{fin}} \subseteq (0, a] \subseteq W_0$ be a subset of forbidden values in W_0 and $F_{\text{inf}} \subseteq (a, b] \subseteq W_0$ such that the set of all forbidden values is $F_{\text{fin}} \cup \left[\bigcup_{i=0}^{\infty} F_{\text{inf}} + i(b-a) \right]$. There exists a finite set X such that $F_{\text{fin}} \cup F_{\text{inf}} \subseteq X \subseteq W_0$ and we can compute the values of X avoiding the values of F in polynomial time in size of W_0 . Symmetrical claim holds if the winning set consists of only negative values.*

PROOF. Let $m = \min(A_0)$ and $M = \max(A_0)$ be the smallest and the largest moves of Adam. Let $X = (m, b + (b-a) + M]$. Clearly $F_{\text{fin}} \cup F_{\text{inf}} \subseteq X$. We can construct a reachability game on a finite arena X by having two copies of the interval X , one for Eve and one for Adam. We connect integers in Eve's (Adam's) interval to integers in Adam's (Eve's) interval corresponding to her (his) moves.

The interval X can be partitioned into three parts, $(-m, a]$, $(a, b]$ and $(b, b+(b-a)+M]$. Intuitively, the first interval $(-m, a]$ corresponds to F_{fin} , the second interval $(a, b]$ to F_{inf} and the final interval to the set $\bigcup_{i=1}^{\infty} F_{\text{inf}} + i(b-a)$. As the finite interval corresponds to several sets, Eve can also move from $x \in (b+m, 2b-a+M]$ to y if there exists a move from $x + (b-a)$ to y . Additionally, if $f \in F_{\text{fin}} \cup F_{\text{inf}}$, then Adam can move to the sink state \top which is losing for Eve. Finally, there exists an edge from a state x if the owner of the state has a move y in the robot game such that $x + y < 0$.

More formally, Adam has states $Q_A = \{\square \times [0, 2b-a]\} \cup \{\top\}$ and Eve has states $Q_E = \{\bigcirc \times [m, 2b-a+M]\}$. The transitions of the game are

$$\begin{aligned} T = & \{((\square, x), (\bigcirc, y)) \in Q_A \times Q_E \mid y - x \in A\} \\ & \cup \{((\bigcirc, x), (\square, y)) \in Q_E \times Q_A \mid y - x \in E\} \\ & \cup \{((\bigcirc, x), (\square, y)) \in Q_E \times Q_A \mid y - (x + b - a) \in E, x \in (b+m, 2b-a+M]\} \\ & \cup \{((\square, x), \top) \mid x \in F_{\text{fin}} \cup F_{\text{inf}} + b - a\} \\ & \cup \{((\bigcirc, x), \top) \in Q_E \times Q_A \mid \exists e \in E, x + e < 0\} \cup \{(\top, \top)\}. \end{aligned}$$

Eve wins the game if she can reach $(\square, 0)$. The winning values of this game can be computed using the attractor construction in polynomial time [29]. \square

Example 7. Let $(\{(t, -1, t)\}), (\{(s, -1, s), (s, -2, s)\})$ be a robot game. Let $F = \{3\} \cup \{x \in \mathbb{Z}^+ \mid x \equiv 2 \pmod{3}, x > 2\}$ be the set of forbidden values. By Lemma 6 we can construct a reachability game G depicted in Figure 13.

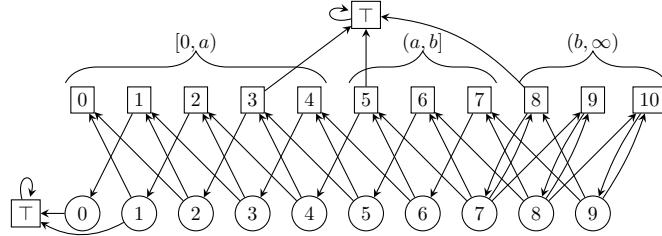


Figure 13: A reachability game on finite arena constructed from a robot game and a set of forbidden values.

Now we are ready to extend Adam's state structure to flat graphs. The algorithm is essentially the same as the one described previously. We utilise the topological sorting to remove forbidden points from the winning sets starting from the end of the graph using Lemma 5. Then we construct the set of avoidable values using Lemma 6.

Theorem 8. One-dimensional flat robot games with states are EXPTIME-complete.

PROOF. Let (A, E) be a FRGS where Adam has k states, t_1, \dots, t_k , such that $(t_i, z, t_j) \in A$ only if $i \leq j$. Denote by $A_i = \{(t_i, z, t_i) \in A\}$. Using the algorithm of [20], we compute the winning set for each pair (A_i, E) . Then, starting from k , we compute sets of forbidden values using Lemma 5. After computing the forbidden values, we compute the avoidable values using Lemma 6. Finally, we update the sets of winning values using the forbidden and avoidable values. \square

7. Conclusion

Robot games are subfamily of counter reachability games where the game is played on a graph with vertices partitioned between players. It has been proved that deciding the winner in two-dimensional counter reachability games is undecidable [12]. Our result can be seen as strengthening of this as our arena is a graph without self-loops and with one vertex for each player, i.e., both players are stateless.

In [9] and [11], VASS games, where the game is played on a graph and counters are always positive, were considered. It was proven that already in two dimensions it is undecidable who wins if Eve's goal is to reach a particular vertex with counter $(0, 0)$. On the other hand, if it can be any vertex, then the problem is $(k - 1)$ -EXPTIME for a game with k counters. Later, the result was improved to PTIME for $k = 2$ [30]. In [31], the authors improved the upper bound for the problem to $2 - \text{EXPTIME}$ if the dimension is not fixed and provided a pseudo-polynomial time algorithm if the dimension is fixed. In [12], the possible counter values were extended to all integers and it was proven that the problem remains undecidable. Hunter considered the variants of games, where updates on the counters are done in binary, and showed that one-dimensional games are EXPSPACE-complete [28]. While these games have reachability objectives, it is also

possible to extend the objectives of the games to energy constraints [7] or parity constraints [6, 8, 32]. Recently, in [33, 34] the safety problem was considered for Minkowski games. As authors point out, robot games are a special case of Minkowski games and while the safety problem is undecidable for Minkowski games, it is decidable for robot games.

The proofs of undecidability of VASS games and counter reachability in two dimensions in [13, 11] use the state structure of the game to embed the state structure of a 2-counter machine. Also in our result on robot games with states, Eve simulates the state transitions of a 2-counter machine with her underlying automaton. On the other hand, the stateless game is essentially different as we have to represent state transitions with integers. When simulating a two-counter machine, it is possible for Eve to make a wrong move and then Adam is able to ensure his victory from this point onward. In robot games, Eve's state is dependent only on her previous moves, while in VASS games or counter reachability games, Adam's moves effect which state Eve enters. Because of this, Adam's cheat catching ability is implemented in a different way.

The construction of robot games with states was first presented in the PhD thesis of one of the authors, [21], where it was also proved that robot games in dimension three are undecidable. The undecidability of 2RG is proved by a new technique of embedding state transitions of a 2CM into integers. It would be interesting to see whether the same approach can be applied to other automata and games, such as stateless VASS games. In this paper we present very minimalistic model that is possible to use as a reference model and to prove undecidability results in more complex games. Recently our result has been already used in [35], where the authors proved that multidimensional average-energy games are undecidable by reducing them to robot games in dimension two.

Korec showed in [36] that there exists a universal Minsky machine with 32 instructions. The natural question of a universal game arises: Is it possible to construct a fixed robot game simulating a universal 2CM? This game would have fixed moves and only the initial vector would affect the result. In [37], it was proven that two-dimensional robot games where both players have two moves are decidable in polynomial time. Consider the machine with 32 instructions. We can construct a robot game from it and count the number of moves. Thus it is undecidable whether Eve has a winning strategy in a two-dimensional robot game where Eve has at least 2083 moves and Adam has 8 moves.

We proved that one-dimensional robot games with states are EXPSPACE-complete. In our construction Eve had states, while Adam was stateless. Motivated by this, we considered games where Adam had states and Eve was stateless. When limiting Adam's state structure to flat automata, we showed that the games are EXPTIME-complete.

In the future, it would be interesting to see whether non-flatness is the property that increases the complexity of deciding the winner from EXPTIME of robot games to EXPSPACE of robot games with states. In particular, whether the complexity of deciding the winner of a robot game, where Adam is stateless and Eve has flat state structure, is EXPTIME-complete or EXPSPACE-complete. Similarly, the tight complexity of robot games with states, where Adam has an arbitrary state structure and Eve is stateless, remains open.

References

- [1] R. Niskanen, I. Potapov, J. Reichert, Undecidability of two-dimensional robot games, in: Proceedings of MFCS 2016, Vol. 58 of LIPIcs, 2016, pp. 73:1–73:13. doi:10.4230/LIPIcs.MFCS.2016.73.

- [2] R. Niskanen, Robot games with states in dimension one, in: Proceedings of RP 2016, Vol. 9899 of LNCS, 2016, pp. 163–176. doi:[10.1007/978-3-319-45994-3_12](https://doi.org/10.1007/978-3-319-45994-3_12).
- [3] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (5) (2002) 672–713. doi:[10.1145/585265.585270](https://doi.org/10.1145/585265.585270).
- [4] O. Kupferman, M. Y. Vardi, P. Wolper, An automata-theoretic approach to branching-time model checking, J. ACM 47 (2) (2000) 312–360. doi:[10.1145/333979.333987](https://doi.org/10.1145/333979.333987).
- [5] I. Walukiewicz, Pushdown processes: Games and model-checking, Inf. Comput. 164 (2) (2001) 234–263. doi:[10.1006/inco.2000.2894](https://doi.org/10.1006/inco.2000.2894).
- [6] P. A. Abdulla, R. Mayr, A. Sangnier, J. Sproston, Solving parity games on integer vectors, in: Proceedings of CONCUR 2013, Vol. 8052 of LNCS, 2013, pp. 106–120. doi:[10.1007/978-3-642-40184-8_9](https://doi.org/10.1007/978-3-642-40184-8_9).
- [7] U. Fahrenberg, L. Juhl, K. G. Larsen, J. Srba, Energy games in multiweighted automata, in: Proceedings of ICTAC 2011, Vol. 6916 of LNCS, 2011, pp. 95–115. doi:[10.1007/978-3-642-23283-1_9](https://doi.org/10.1007/978-3-642-23283-1_9).
- [8] K. Chatterjee, L. Doyen, Energy parity games, Theor. Comput. Sci. 458 (2012) 49–60. doi:[10.1016/j.tcs.2012.07.038](https://doi.org/10.1016/j.tcs.2012.07.038).
- [9] P. A. Abdulla, A. Bouajjani, J. d’Orso, Monotonic and downward closed games, J. Log. Comput. 18 (1) (2008) 153–169. doi:[10.1093/logcom/exm062](https://doi.org/10.1093/logcom/exm062).
- [10] T. Brázdil, V. Brozek, K. Etessami, One-counter stochastic games, in: In proceedings of FSTTCS 2010, Vol. 8 of LIPIcs, 2010, pp. 108–119. doi:[10.4230/LIPIcs.FSTTCS.2010.108](https://doi.org/10.4230/LIPIcs.FSTTCS.2010.108).
- [11] T. Brázdil, P. Jančar, A. Kučera, Reachability games on extended vector addition systems with states, in: Proceedings of ICALP 2010, Vol. 6199 of LNCS, 2010, pp. 478–489. doi:[10.1007/978-3-642-14162-1_40](https://doi.org/10.1007/978-3-642-14162-1_40).
- [12] J. Reichert, On the complexity of counter reachability games, Fundam. Inform. 143 (3-4) (2016) 415–436. doi:[10.3233/FI-2016-1320](https://doi.org/10.3233/FI-2016-1320).
- [13] P. A. Abdulla, A. Bouajjani, J. d’Orso, Deciding monotonic games, in: Proceedings of CSL 2003, Vol. 2803 of LNCS, 2003, pp. 1–14. doi:[10.1007/978-3-540-45220-1_1](https://doi.org/10.1007/978-3-540-45220-1_1).
- [14] L. Doyen, A. Rabinovich, Robot games. Personal website, 2011, Tech. Rep. LSV-13-02, LSV, ENS Cachan (2013).
URL <http://www.lsv.ens-cachan.fr/Publis/RAPPORTS%5FLSV/PDF/rr-lsv-2013-02.pdf>
- [15] C. Haase, S. Halfon, Integer vector addition systems with states, in: Proceedings of RP 2014, Vol. 8762 of LNCS, 2014, pp. 112–124. doi:[10.1007/978-3-319-11439-2_9](https://doi.org/10.1007/978-3-319-11439-2_9).
- [16] M. Blondin, A. Finkel, S. Göller, C. Haase, P. McKenzie, Reachability in two-dimensional vector addition systems with states is PSPACE-complete, in: Proceedings of LICS 2015, 2015, pp. 32–43. doi:[10.1109/LICS.2015.14](https://doi.org/10.1109/LICS.2015.14).
- [17] M. Blondin, A. Finkel, P. McKenzie, Well behaved transition systems, Logical Methods in Computer Science ; Volume 13 (2017) Issue 3 ; 1860–5974doi:[10.23638/lmcs-13\(3:24\)2017](https://doi.org/10.23638/lmcs-13(3:24)2017).
- [18] M. Blondin, C. Haase, F. Mazowiecki, Affine extensions of integer vector addition systems with states, in: Proceedings of CONCUR 2018, Vol. 118 of LIPIcs, Schloss Dagstuhl, 2018, pp. 14:1–14:17. doi:[10.4230/lipics.concur.2018.14](https://doi.org/10.4230/lipics.concur.2018.14).
- [19] D. A. Martin, Borel determinacy, Annals of Mathematics 102 (2) (1975) 363–371. doi:[10.2307/1971035](https://doi.org/10.2307/1971035).
- [20] A. Arul, J. Reichert, The complexity of robot games on the integer line, in: Proceedings of QAPL 2013, Vol. 117 of EPTCS, 2013, pp. 132–148. doi:[10.4204/EPTCS.117.9](https://doi.org/10.4204/EPTCS.117.9).
- [21] J. Reichert, Reachability games with counters: Decidability and algorithms, Doctoral thesis, Laboratoire Spécification et Vérification, ENS Cachan, France (2015).
- [22] H. Comon, V. Cortier, Flatness is not a weakness, in: Proceedings of CSL 2000, Vol. 1862 of LNCS, 2000, pp. 262–276. doi:[10.1007/3-540-44622-2_17](https://doi.org/10.1007/3-540-44622-2_17).
- [23] H. Comon, Y. Jurski, Multiple counters automata, safety analysis and Presburger arithmetic, in: Proceedings of CAV, 1998, Vol. 1427 of LNCS, 1998, pp. 268–279. doi:[10.1007/BFb0028751](https://doi.org/10.1007/BFb0028751).
- [24] H. Comon, Y. Jurski, Timed automata and the theory of real numbers, in: Proceedings of CONCUR 1999, Vol. 1664 of LNCS, 1999, pp. 242–257. doi:[10.1007/3-540-48320-9_18](https://doi.org/10.1007/3-540-48320-9_18).
- [25] J. Leroux, V. Penelle, G. Sutre, The context-freeness problem is coNP-complete for flat counter systems, in: Proceedings of ATVA 2014, Vol. 8837 of LNCS, 2014, pp. 248–263. doi:[10.1007/978-3-319-11936-6_19](https://doi.org/10.1007/978-3-319-11936-6_19).
- [26] J. Leroux, G. Sutre, Flat counter automata almost everywhere!, in: Proceedings of ATVA 2005, Vol. 3707 of LNCS, 2005, pp. 489–503. doi:[10.1007/11562948_36](https://doi.org/10.1007/11562948_36).
- [27] M. L. Minsky, Computation: finite and infinite machines, Prentice-Hall, Inc., 1967.
- [28] P. Hunter, Reachability in succinct one-counter games, in: Proceedings of RP 2015, Vol. 9328 of LNCS, 2015, pp. 37–49. doi:[10.1007/978-3-319-24537-9_5](https://doi.org/10.1007/978-3-319-24537-9_5).
- [29] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current

- Research, Vol. 2500 of LNCS, Springer, 2002.
- [30] J. Chaloupka, Z-reachability problem for games on 2-dimensional vector addition systems with states in P, *Fundam. Inform.* 123 (1) (2013) 15–42. doi:[10.3233/FI-2013-798](https://doi.org/10.3233/FI-2013-798).
 - [31] M. Jurdzinski, R. Lazic, S. Schmitz, Fixed-dimensional energy games are in pseudo-polynomial time, in: *Proceedings of ICALP 2015*, Vol. 9135 of LNCS, Springer, 2015, pp. 260–272. doi:[10.1007/978-3-662-47666-6_21](https://doi.org/10.1007/978-3-662-47666-6_21).
 - [32] T. Colcombet, M. Jurdzinski, R. Lazic, S. Schmitz, Perfect half space games, in: *Proceedings of LICS 2017*, IEEE Computer Society, 2017, pp. 1–11. doi:[10.1109/LICS.2017.8005105](https://doi.org/10.1109/LICS.2017.8005105).
 - [33] S. L. Roux, A. Pauly, J. Raskin, Minkowski games, in: *STACS 2017*, Vol. 66 of LIPIcs, 2017, pp. 50:1–50:13. doi:[10.4230/LIPIcs.STACS.2017.50](https://doi.org/10.4230/LIPIcs.STACS.2017.50).
 - [34] S. L. Roux, A. Pauly, J.-F. Raskin, Minkowski games, *ACM Transactions on Computational Logic* 19 (3) (2018) 1–29. doi:[10.1145/3230741](https://doi.org/10.1145/3230741).
 - [35] P. Bouyer, P. Hofman, N. Markey, M. Randour, M. Zimmermann, Bounding average-energy games, in: *Foundations of Software Science and Computation Structures*, LNCS, Springer, 2017, pp. 179–195. doi:[10.1007/978-3-662-54458-7_11](https://doi.org/10.1007/978-3-662-54458-7_11).
 - [36] I. Korec, Small universal register machines, *Theor. Comput. Sci.* 168 (2) (1996) 267–301. doi:[10.1016/S0304-3975\(96\)00080-1](https://doi.org/10.1016/S0304-3975(96)00080-1).
 - [37] V. Halava, R. Niskanen, I. Potapov, On robot games of degree two, in: *Proceedings of LATA 2015*, Vol. 8977 of LNCS, 2015, pp. 224–236. doi:[10.1007/978-3-319-15579-1_17](https://doi.org/10.1007/978-3-319-15579-1_17).