Undecidability of Two-dimensional Robot Games

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

1 **Department of Computer Science, University of Liverpool, UK**
{r.niskanen,potapov}@liverpool.ac.uk
2 **LSV, ENS Cachan, France**
reichert@crans.org

Abstract

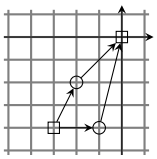
Robot game is a two-player vector addition game played on the integer lattice $\mathbb{Z}^n$. Both players have sets of vectors and in each turn the vector chosen by a player is added to the current configuration vector of the game. One of the players, called Eve, tries to play the game from the initial configuration to the origin while the other player, Adam, tries to avoid the origin. The problem is to decide whether or not Eve has a winning strategy. In this paper we prove undecidability of the robot game in dimension two answering the question formulated by Doyen and Rabinovich in 2011 and closing the gap between undecidable and decidable cases.

Keywords and phrases reachability games, vector addition game, decidability, winning strategy

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. Computational games provide a good framework to model interactive processes and the extensions of classical reachability problems to game schemes, studied in different contexts and settings, have recently garnered considerable interest [2, 3, 5, 6, 8, 10, 16].

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)[1] [1, 6], Doyen and Rabinovich formulated an open problem about the simplest version of games (*robot games*) for which the decidability was unknown [9]. 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 below.



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

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

We say that Eve has a winning strategy if she eventually can reach the zero vector replying on any Adam move and Adam has a winning strategy otherwise. Thus a winning strategy gives

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

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 is EXPTIME-complete [4].

In this 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, we construct a robot game with states that follows the computation of $\mathcal{M}$. As the game lacks the ability to perform directly zero checks, instead Adam has a move allowing him to check whether a counter is positive or not leading, deterministically, either to his 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.

Apart from the solution of the open problem, the main contribution of this paper 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.

Previous research: 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 [16]. 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 [2] and [6], 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 the 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 $k = 2$ [7]. In [16], 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 [13]. While these games have reachability objectives, it is also possible to extend the objectives of the games to energy constrains [10] or parity constrains [3, 8].

The proofs of undecidability of VASS games and counter reachability in two dimensions in [1, 6] use the state structure of the game to embed the state structure of a 2-counter machine. In this sense, our result on robot games with states is comparable, as 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 with states Adam's cheat catching ability is different from when a game is played on a graph. Also 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.

2 Notation and Definitions

We denote the set of all integers by $\mathbb{Z}$ and the set of all non-negative integers by $\mathbb{N}$. By 0_n we denote a n-dimensional zero vector.

A *counter reachability game* (CRG) consists of a directed graph $G = (V, E)$, where the set of vertices is partitioned into two parts, V_1 and V_2, each edge $e \in E \subseteq V \times \mathbb{Z}^n \times V$ is labelled with vectors in $\mathbb{Z}^n$, and an initial vector $\mathbf{x}_0 \in \mathbb{Z}^n$. A *configuration* of the game is $(v, \mathbf{x})$, a successive configuration is $(v', \mathbf{x} + \mathbf{x}')$, where an edge $(v, \mathbf{x}', v') \in E$ is chosen by player 1 if $v \in V_1$ or by player 2 if $v \in V_2$. 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 $v_f \in V$ 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 ◯ for Eve's states and □ for Adam's states.

A *robot game* (RG) [9] is a special case of the counter reachability games, where the graph consists of only two vertices, q_0 of Adam and q of Eve. The goal of the game is the configuration $(q_0, 0_n)$. 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 games, 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 $\mathbf{x}_0$. The problem is EXPTIME-complete in dimension one [4] and was open for dimension two.

An extension of robot games where players have control states is called *robot games with states* (RGS). We consider only the games where Adam's state structure is trivial, i.e., he has only one state and all moves are self-loops. RGS consists of (A, E) where A is a finite subset of $\mathbb{Z}^n$ that Adam can apply during his turn and E is a finite subset of $V \times \mathbb{Z}^n \times V$ of Eve. The configuration is now a pair $(s, \mathbf{v})$ consisting of Eve's control state s and a counter vector $\mathbf{v} \in \mathbb{Z}^n$. Eve updates her control state when she makes a move: in the configuration $(s, \mathbf{v})$, for any vector $\mathbf{v}$, only moves of the form $(s, \mathbf{x}, t)$ are enabled, and with one such move the new configuration is $(t, \mathbf{v} + \mathbf{x})$. Eve wins if, and only if, after her turn, the configuration is $(s, (0, 0))$ for any $s \in V$. The decision problem associated with robot games with states asks whether Eve has a winning strategy from a given configuration.

A *Minsky machine*, introduced in [15], 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 $(s, (y, z)) \in Q \times \mathbb{N}^2$, 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 [15]. Another well-known undecidable problem for 2CM is whether a configuration where both counters are zero is reachable. The undecidability follows from the halting problem by modifying a 2CM to ensure that both counters are zero only in the halting state; see for example [17] 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 $Q \times \{(0, 0)\}$ 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. The game lacks the ability to perform zero checks present in two-counter machines, instead 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 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 1. 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 $(q, (0, 0))$ for some $q \in Q$ 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 $(q, (0, 0))$ for some $q \in Q$ (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).
- This leads to the possibility that 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).

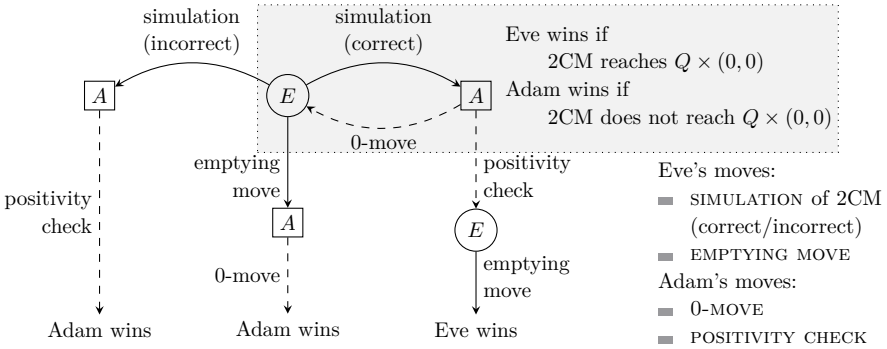


Figure 1 Progress of 2RGS

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 $a, b \in \{0, +\}$ 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

$$\{(s_{ab}, c_1{+}{+}, t_{+b}) \mid (s, c_1{+}{+}, t) \in T, a, b \in \{0, +\}\}, \{(s_{ab}, c_2{+}{+}, t_{a+}) \mid (s, c_2{+}{+}, t) \in T, a, b \in \{0, +\}\},$$
$$\{(s_{+b}, c_1{-}{-}, t_{ab}) \mid (s, c_1{-}{-}, t) \in T, a, b \in \{0, +\}\}, \{(s_{a+}, c_2{-}{-}, t_{ab}) \mid (s, c_2{-}{-}, t) \in T, a, b \in \{0, +\}\},$$
$$\{(s_{0b}, c_1{=}{=}0, t_{0b}) \mid (s, c_1{=}{=}0, t) \in T, b \in \{0, +\}\}, \{(s_{a0}, c_2{=}{=}0, t_{a0}) \mid (s, c_2{=}{=}0, t) \in T, a \in \{0, +\}\}.$$

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

counter value	flag		flag	
$c_i > 1$	$+$	$\to$	$+$	**correct flag**
$c_i > 1$	$+$	$\to$	0	wrong flag
$c_i = 1$	$+$	$\to$	$+$	wrong flag
$c_i = 1$	$+$	$\to$	0	**correct flag**

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, see Figure 6 in the Appendix:

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

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

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 2.

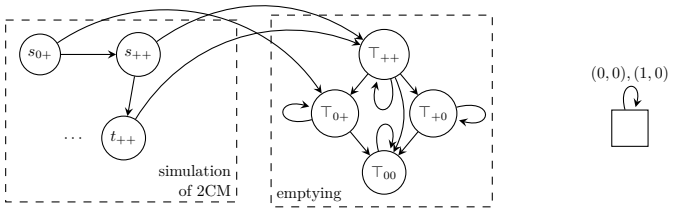


Figure 2 An illustration of state transitions of Eve and Adam

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 $(s_{\overline{a}\overline{b}}, (y,z))$ and we define $\overline{a} = +, \overline{b} = 0$ if $y = 1$ and $\overline{a} = 0, \overline{b} = +$ if $y = 0$. The initial configuration in the robot game with states is then $(s_{\overline{a}\overline{b}}, (4y, z))$. The effect of simulating moves, emptying moves and positivity check modulo four is depicted in Figure 3.

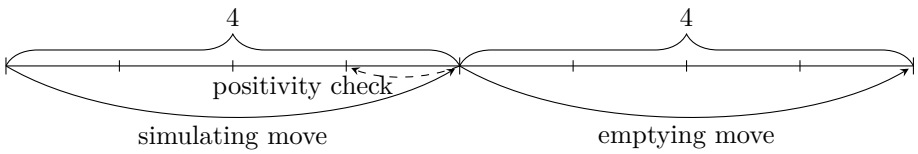


Figure 3 An illustration of changes in an interval when simulating or emptying moves of Eve or positivity check of Adam is applied

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 easy to see that correct moves of Eve simulate the 2CM and that a configuration $(s, (0, 0))$ of the 2CM is reachable if and only if it 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), Adam has a winning strategy starting with the* POSITIVITY CHECK*.*

Proof. Assume that Eve made a mistake regarding the positivity of the first counter. As noted previously, there are two ways she can make a mistake. Either the configuration is $(s_{0b}, (4x, y))$, where $x \geq 1$ or $(s_{+b}, (0, y))$. In both cases Adam plays his POSITIVITY CHECK which changes the parity of the first counter. That is, after Adam's turn, the first counter is 1 (mod 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, he will play his POSITIVITY CHECK if and only if the first counter is not 3 (mod 4). Eve cannot make the counter 0, as she cannot even make it 0 (mod 4). Thus Eve has to play a move adding -1 to the first counter. The only move for that is $(s_{ab}, (-1, 0), \top_{ab})$ which takes Eve to an emptying state. In the first case the emptying state is $\top_{0b}$ and all the transitions from it do not modify the first counter, i.e., Eve cannot reach $(0, 0)$. In the second case 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 symmetric and is proven analogously. ◀

▶ **Lemma 5.** *If Eve plays only correct* SIMULATING MOVES *before Adam plays the* POSITIVITY CHECK *for the first time, 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 (mod 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 (mod 4). As in 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 2-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 1 and 4. 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 2-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 4, we present a schematic similar to Figure 1 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).
- This leads to the possibility that 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).

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

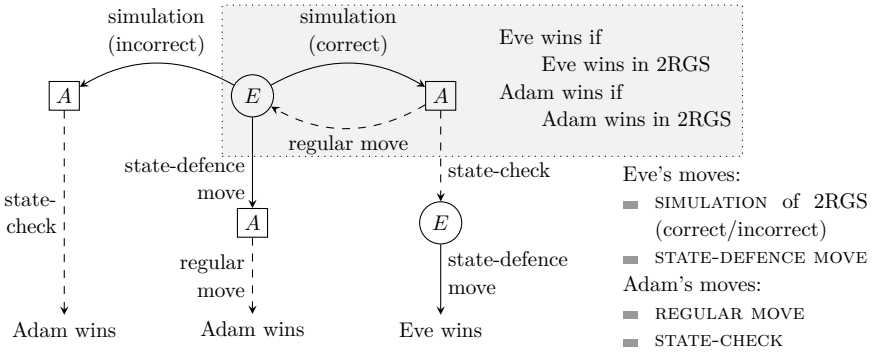


Figure 4 Progress of 2RG

s_j, $-8^i + 8^j$ is added to the second counter. Let $(s_i, (x, y))$ be a configuration in a two-dimensional robot game with m states. Let us represent the state s_i with m-dimensional characteristic vector $\mathbf{s_i} = (s_1, \ldots, s_m)$ where s_i is 1 and $s_j = 0$ for all $j \neq i$. We can now map $\mathbf{s_i}$ to an integer defined by the sum $\sum_{k=1}^{m} s_k 8^k$. A transition from s_i to s_j can be simulated by adding $\sum_{k=1}^{m} s_k 8^k$, where $s_k = -1$ if $k = i$, $s_k = 1$ if $k = j$, and zero otherwise. Note that 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 this 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 5. 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 $(s_i, (c_1, c_2)) \mapsto (c_1, c_2 \cdot 4 \cdot 8^n + 8^i)$.

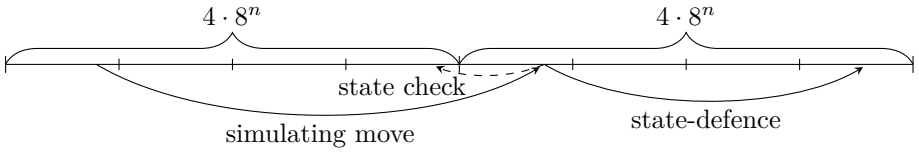


Figure 5 An illustration of changes in interval when simulating or state-defence moves of Eve or state check of Adam is applied

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. Let Q be the set of states of Eve in 2RGS. We create an emptying gadget for Eve similar to the one 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 a 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 $\mathcal{T} = \{\top_{++}, \top'_{++}, \top_{0+}, \top'_{0+}, \top_{+0}, \top'_{+0}\}$. We think of elements of $Q \cup \mathcal{T} \cup \{\top_{00}\}$ as integers in $\{0, \ldots, n-1\}$ 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:

$$\textsc{Add}(1,x) := (x,0); \qquad \textsc{Move}(j,k) := (0, -8^j + 8^k), \text{ for } 0 \le j,k \le n-1;$$
$$\textsc{Add}(2,x) := (0, 4x \cdot 8^n); \quad \textsc{Check}(i) := (0, -5 \cdot 8^i - 8^n), \text{ for } n-6 \le i \le n-1.$$

The initial vector of the robot game is $\textsc{Add}(1,x) + \textsc{Add}(2,y) + \textsc{Move}(\top_{00}, s)$, 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 $\textsc{Add}(1,-1)$, $\textsc{Add}(2,1)$, $\textsc{Move}(s,t)$, $\textsc{Check}(8)$ in succession:

$$\overbrace{(1, 0 \cdot 4 \cdot 8^9}^{\text{2RGS counters}} + \overbrace{0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3}^{\mathcal{T}} + \overbrace{0 \cdot 8^2 + 1 \cdot 8^1}^{\text{states of 2RGS}} \overbrace{-1 \cdot 8^0)}^{\top_{00}} \xrightarrow{\textsc{Add}(1,-1)}$$
$$(0, 0 \cdot 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \xrightarrow{\textsc{Add}(2,1)}$$
$$(0, 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 - 1 \cdot 8^0) \xrightarrow{\textsc{Move}(s,t)}$$
$$(0, 4 \cdot 8^9 + 0 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0) \xrightarrow{\textsc{Check}(8)}$$
$$(0, 3 \cdot 8^9 - 5 \cdot 8^8 + 0 \cdot 8^7 + 0 \cdot 8^6 + 0 \cdot 8^5 + 0 \cdot 8^4 + 0 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 - 1 \cdot 8^0).$$

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, $\{\textsc{Check}(i) \mid i \in \mathcal{T}\}$. 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 $\textsc{Add}(1,x) + \textsc{Add}(2,y) + \textsc{Move}(s,t) = (x, 4y \cdot 8^n - 8^s + 8^t) \in E$. 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 $\{\textsc{Move}(s_{00}, \top_{00}) - \alpha \mid \alpha \in A_1\}$. 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 $\{\textsc{Move}(s_{ab}, k) - \textsc{Check}(i) \mid (a,b) \in \{0,+\}^2 \setminus \{(0,0)\}, k \in \{\top_{ab}, \top'_{ab}\}, k \neq i, i \in \mathcal{T}\}$. For s_{00}, Eve has a move $\{\textsc{Move}(s_{00}, \top_{00}) - \textsc{Check}(i) \mid i \in \mathcal{T}\}$.

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

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 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 $\textsc{Move}(s_{00}, \top_{00}) - \alpha$, 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, 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 $\textsc{Check}(i)$), 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 $\textsc{Check}(i)$ forcing Eve to play a move containing $-\textsc{Check}(i)$ 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 $\textsc{Move}(s,t)$ making the coefficient of 8^s negative, which has been covered previously. ◀

▶ **Lemma 12.** *If Eve plays only correct* REGULAR MOVES *until Adam plays a* STATE-CHECK *for the first time, 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 $\textsc{Move}(s_{ab},k) - \textsc{Check}(i)$ where s_{ab} is the non-zero coefficient, $\textsc{Check}(i)$ 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 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 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. It is possible to apply it to matrix games introduced in [11] to show undecidability in $\mathbb{Z}^{3\times 3}$; see the proof in the Appendix.

Final remarks: The construction of robot games with states was first presented in the PhD thesis of one of the authors, [17], 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. It is not clear how to embed not only state transitions of an automaton, but also the input word.

Korec showed in [14] 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 [12], 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.

References

1 Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Deciding monotonic games. In *Proceedings of CSL 2003*, volume 2803 of *LNCS*, pages 1–14, 2003.

2 Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Monotonic and downward closed games. *J. Log. Comput.*, 18(1):153–169, 2008.

3 Parosh Aziz Abdulla, Richard Mayr, Arnaud Sangnier, and Jeremy Sproston. Solving parity games on integer vectors. In *Proceedings of CONCUR 2013*, volume 8052 of *LNCS*, pages 106–120, 2013.

4 Arjun Arul and Julien Reichert. The complexity of robot games on the integer line. In *Proceedings of QAPL 2013*, volume 117 of *EPTCS*, pages 132–148, 2013.

5 Tomás Brázdil, Václav Brozek, and Kousha Etessami. One-counter stochastic games. In *In proceedings of FSTTCS 2010*, volume 8 of *LIPIcs*, pages 108–119, 2010.

6 Tomáš Brázdil, Petr Jančar, and Antonín Kučera. Reachability games on extended vector addition systems with states. In *Proc. of ICALP 2010*, volume 6199 of *LNCS*, pages 478–489, 2010.

7 Jakub Chaloupka. Z-reachability problem for games on 2-dimensional vector addition systems with states is in P. *Fundam. Inform.*, 123(1):15–42, 2013.

8 Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theor. Comput. Sci.*, 458:49–60, 2012.

9 Laurent Doyen and Alexander Rabinovich. Robot games. Personal website, 2011. Technical Report LSV-13-02, `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2013-02.pdf`, LSV, ENS Cachan, 2013.

10 Uli Fahrenberg, Line Juhl, Kim G. Larsen, and Jirí Srba. Energy games in multiweighted automata. In *Proceedings of ICTAC 2011*, volume 6916 of *LNCS*, pages 95–115, 2011.

11 Vesa Halava, Tero Harju, Reino Niskanen, and Igor Potapov. Weighted automata on infinite words in the context of attacker-defender games. In *Proceedings of CiE 2015*, volume 9136 of *LNCS*, pages 206–215, 2015.

12 Vesa Halava, Reino Niskanen, and Igor Potapov. On robot games of degree two. In *Proceedings of LATA 2015*, volume 8977 of *LNCS*, pages 224–236, 2015.

13 Paul Hunter. Reachability in succinct one-counter games. In *Proceedings of RP 2015*, volume 9328 of *LNCS*, pages 37–49, 2015.

14 Ivan Korec. Small universal register machines. *Theor. Comput. Sci.*, 168(2):267–301, 1996.

15 Marvin L Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

16 Julien Reichert. On the complexity of counter reachability games. In *Proceedings of RP 2013*, volume 8169 of *LNCS*, pages 196–208, 2013.

17 Julien Reichert. *Reachability Games with Counters: Decidability and Algorithms*. Doctoral thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, July 2015.

A Appendix

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

Proof. Let (Q,T) be a deterministic two-counter Minsky machine. We build (Q',T') such that the run in (Q,T) reaches a configuration in $\{\perp\} \times \mathbb{N}^2$ if, and only if, the run in (Q',T') returns to a configuration in $Q \times \{(0,0)\}$. To do this, we shift the values of (Q',T') to ensure that at no point both counters are zero and add emptying states used to empty the counters before reaching the halting state $\perp$. That is, a configuration $(q,(0,0))$ is reachable if and only if $q = \perp$ is reachable. Zero-checks and decrementing are implemented by shifting the counter value back down and then performing the zero-check after which the counter value is shifted back up. It follows that $(0,0)$ is reachable in (Q',T') if and only if (Q,T) halts. ◀

▶ **Corollary 15.** *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.*

Proof. We count the number of moves each player has in the robot game constructed in the previous section. Adam has 8 moves and Eve has at most $58m + 227$ moves, where m is the number of states in the original 2-counter machine. Korec showed in [14] that there is a universal 2-counter machine with 32 instructions, which means that it has at most 32 states. Thus there are at most 2083 moves for Eve. ◀

Matrix games: We apply the result on two-dimensional robot games to matrix games introduced in [11]. A *matrix game on vectors* (or *matrix game* for short) consists of two players, Eve and Adam, having sets of linear transformations $\{U_1,\dots,U_r\} \subseteq \mathbb{Z}^{n\times n}$ and $\{V_1,\dots,V_s\} \subseteq \mathbb{Z}^{n\times n}$ respectively, an *initial vector* $\mathbf{x}_0 \in \mathbb{Z}^n$ of the game representing the starting position, and a *target vector* $\mathbf{y} \in \mathbb{Z}^n$. Starting from $\mathbf{x}_0$, players move the current point by applying available linear transformations (by matrix multiplication) from their respective sets in turns. The decision problem of the matrix game is to check whether there exist a winning strategy for Eve to reach the target from the starting point (vectors in $\mathbb{Z}^n$) of the game. Note that in our formulation the vectors are horizontal and players multiply it from the right.

In [11], it was proven that the game is undecidable starting from dimension four. By encoding 2-dimensional robot game into matrices, we get a matrix game of dimension three for which it is undecidable whether Eve has a winning strategy.

▶ **Theorem 16.** *It is undecidable whether Eve has a winning strategy in three-dimensional matrix games.*

Proof. Let (A,E) be a two-dimensional robot game with initial vector (x_0,y_0). For each move (x,y) of Eve (Adam) in robot game, Eve (Adam) has a respective move $\left(\begin{smallmatrix}1&0&0\\x&1&y\\0&0&1\end{smallmatrix}\right)$. The initial vector of matrix game is $(x_0,1,y_0)$ and the target is $(0,1,0)$. It is easy to see that adding a vector (x,y) to a vector (u,v) in robot games corresponds to matrix multiplication

$$(u,1,v)\begin{pmatrix}1&0&0\\x&1&y\\0&0&1\end{pmatrix} = (u+x,1,y+v).$$

It is easy to see that, in the matrix game, Eve has a winning strategy to reach $(0,1,0)$ if and only if she has a winning strategy to reach $(0,0)$ in the robot game. Since the latter problem

is undecidable, so is deciding whether Eve has a winning strategy in three-dimensional matrix games. ◀

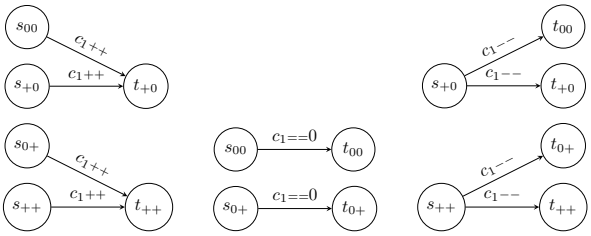


Figure 6 Transitions modifying the first counter in the modified 2CM