WinPop: Making Populations Win Together

Nathalie Bertrand
Univ Rennes, Inria, CNRS, IRISA, France

Patricia Bouyer
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190 Gif-sur-Yvette, France

Luc Lapointe
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190 Gif-sur-Yvette, France

Corto Mascle
MPI-SWS, Kaiserslautern, Germany

Abstract

We consider a novel graph-based problem, in which a population of arbitrary size aims at achieving a common objective. More specifically, WinPop is a synthesis problem defined by a finite graph with edges labels in { ✔ , − , ✘ }. The instance is positive if there exists a sequence $(\pi_i)_{i\in\mathbb{N}}$ of infinite paths such that for any fixed population size $N \in \mathbb{N}_{>0}$, there is a path whose N-th transition is labelled ✔ and all previous paths have their N-th transition labelled by − .

Alternatively, WinPop can also be cast as a 2D-tiling problem with vertical and horizontal constraints: the horizontal constraint reflects the possible paths in the input graph, and the vertical one encodes that a ✔ -label eventually occurs, before any ✘ -label. Finally, WinPop also corresponds to the existence of a coalition strategy for a reachability objective in parameterized concurrent games.

We use algebraic tools to show that the problem can be solved in polynomial space. First we exhibit a finite semigroup whose elements summarize coalition strategies over a finite interval of population sizes. Then, we characterize the existence of winning strategies by the existence of particular elements in this semigroup. Finally, we provide a matching complexity lower bound, to conclude that WinPop is PSPACE-complete.

2012 ACM Subject Classification Theory of computation → Verification by model checking; Theory of computation → Automata over infinite objects

Keywords and phrases Parameterized systems, Automata, Semigroups, Concurrent games, Tiling Problem

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.18

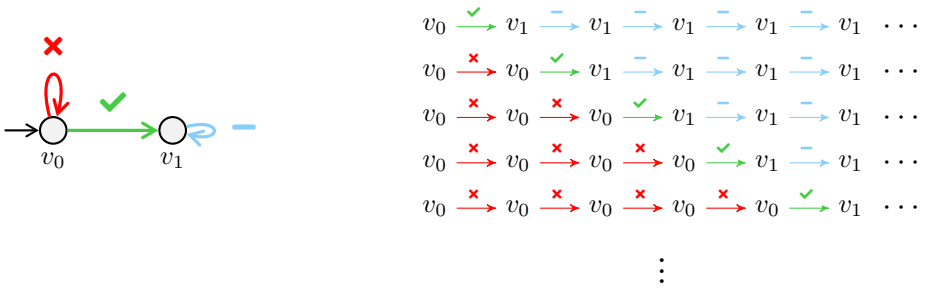
Related Version *Full Version*: `https://arxiv.org/abs/2510.02984` [3]

Funding This work has received funding from ANR BisoUS (ANR-22-CE48-0012) and ANR PaVeDyS (ANR-23-CE48-0005).

1 Introduction

Winning populations. We introduce a novel graph-based synthesis problem, WinPop, whose input is a finite directed graph with a distinguished initial vertex and in which all edges are labelled with a symbol in { ✔ , − , ✘ }. The labelled graph is a positive instance of WinPop if there exists a sequence of initial infinite paths such that, for all $N \in \mathbb{N}_{>0}$ there is a path whose N-th transition is labelled ✔ and all previous paths have their N-th transition labelled by − . For reasons that will become clearer later, we refer to such a sequence of paths as a *winning population strategy*. Deciding the existence of a winning population strategy, and in the positive case, providing a way to synthesize a winning sequence of infinite paths, form the objective of this paper.

As a first illustration, consider the example in Figure 1. To witness that the labelled graph in Figure 1a is a positive instance of WinPop, one can consider the infinite sequence of paths in Figure 1b. Graphically, each row in Figure 1b is a path in the labelled graph, corresponding to a move of the population strategy: the first move is the first row, followed by the second move, etc. Each column of edge labels corresponds to a population size (the first column for a population of size 1, etc) and belongs to the regular language $L_{\text{−}\,\mathsf{U}\,✓} = \text{−}^*✓(✓ + \text{−} + ✗)^\omega$. This constraint on columns expresses that nothing bad (✗) happens before (from top to bottom) something good (✓) eventually does.



(a) Labelled graph. **(b)** Winning population strategy.

Figure 1 Example of a labelled graph (left) for which there is a winning population strategy to the synthesis problem WinPop.

Alternative views. WinPop can also be seen as a tiling problem, asking for the existence of an infinite 2D-grid with regular constraints on its rows and columns. It takes as input a finite automaton over the alphabet { ✓ , ✗ , − }. Rows of the grid should be labels of infinite runs of the automaton, while vertical lines should have sequences of labels in the language $L_{\text{−}\,\mathsf{U}\,✓}$, expressing a "stay safe until you reach" property. We only consider the three-letter alphabet here, but it would be quite easy to generalise our approach to larger alphabets and languages of the form "A until B" where A and B are subsets of letters.

Also, as we will detail later, a class of concurrent parameterized games can be cast into WinPop. Concurrent parameterized games [4, 5] are games on graphs, in which the number of players –also called the size of the population– is arbitrary and initially unknown. In the cooperative scenario, a coalition strategy aims at achieving a common objective independently of the actual number of players. Notably, this coalition strategy should be *uniform*, whatever the population size. It so happens that WinPop is another way of formalizing the existence of a coalition strategy for reachability objective for simple concurrent parameterized games. This relationship, formalized in Section 3.2, was our original motivation for considering WinPop. It also explains the vocabulary we employ in this paper: populations, games, strategies, etc.

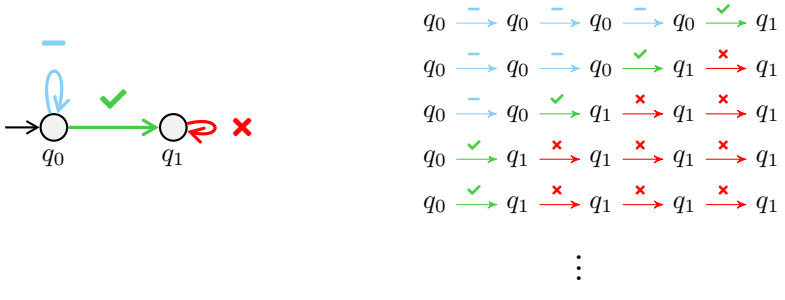
Games with arbitrarily many players. In recent years, several models of games with an arbitrary number of players have been introduced, to model situations in which the number of agents is unknown, or concisely represent infinitely many games instances, one for each number of players. The above-mentioned concurrent parameterized games [4, 5] fall in this category, as well as population control problems [6, 9, 11, 12]. The latter are played on an automaton –or an MDP– and several pebbles move along transitions during a play. For population control, the goal is to drive the pebbles to a common goal, independently of

the population size, *i.e.* for every possible number of pebbles. Similar to parameterized concurrent games, population control fits in the framework of parameterized verification, where the parameter is the number of players.

Tiling and images. Viewing WinPop as a tiling problem connects with other works on images. The expressivity of automata and logics on finite images have been investigated in [15, 13], as well as some extensions of these models to infinite images. Most natural decision problems on these extensions, such as checking emptiness, are either undecidable, or with no decision procedure known yet [10, 1]. For instance, the problem, given two regular languages, to check if there is an image whose rows are all in the first and columns all in the second, is undecidable. This is immediate as one can encode the domino problem [2] constraints (neighbouring tiles should match, vertically and horizontally) in regular languages. In this work, we propose a new technique to obtain decidability of such problems when one of the input languages is from a weaker class, where the domino condition cannot be encoded. A related intriguing open problem is the *lossy tiling problem*, which, given a regular language and a subword-closed one, asks for a family of finite images of unbounded heights whose lines are all in the former and columns in the latter language. It was shown that the satisfiability problem for a quantitative variant of monadic second-order logic on infinite words, called weak asymptotic monadic second-order logic, reduces to this problem [7].

Overview of the paper. Our main contribution is an algorithm to solve the WinPop problem. While tiling problems are well-known to be undecidable in general, we present a new technique which uses the particular shape of our column languages. Specifically, they are closed under a form of stuttering, which is that one can always repeat any character that has been seen before without leaving the language. We show that we can abstract columns by the set of elements appearing in them, and their order of first appearance. The resulting abstractions are called *frontiers*. There is a natural composition operation on these frontiers, written $*$, which gives us abstractions of contiguous sequences of columns rather than single columns. This operation induces a semigroup structure.

While those frontiers let us represent conveniently finite sequences of columns, jumping to infinite sequences requires more work. Intuitively, we need to distinguish instances where we have to reach ✔ lower and lower as we move right in the grid (as in Figure 1, a positive instance) from those where we would have to reach it higher and higher (see Figure 2, a negative instance).



(a) Labelled automaton. **(b)** A grid of finite width.

Figure 2 Example of a labelled graph (left) for which there is no winning population strategy. However, there are grids of arbitrary finite width satisfying the constraints.

Thanks to Ramsey's theorem on infinite graphs, the positive instances of the WinPop problem can be characterized using a structure with a finite prefix and a repeating pattern, abstracted by frontiers (respectively f and g in the theorem below), in our main technical result:

▶ **Theorem 1.** *There is a winning population strategy if and only if there exist two frontiers $f, g \in \psi(0^+)$ such that:*

1. *f is initial,* **2.** *g is ω-iterable,* **3.** *$f * g \to f$,* **4.** *$g * g \to g$.*

The precise notions and notations of this result will be made clear later in the paper. Intuitively, the condition that f is initial ensures that it encodes a finite sequence of columns at the beginning, and the ω-iterability of g guarantees further portions of all moves follow a pattern which can be iterated indefinitely, and the two other conditions impose that f and g combine nicely. Since the number of frontiers is at most exponential in the size of the labelled deterministic automaton, this characterization allows one to decide the problem in polynomial space. We complement this result with a matching complexity lower bound, thereby establishing that the problem is PSPACE-complete.

Due to space constraints, omitted proofs are in the long version of this paper [3].

2 Graph-based population games

2.1 Notations

We write $\mathbb{N}_{>0}$ for the set of positive integers. Given $i \leq j \in \mathbb{N}_{>0}$, $[\![i; j]\!]$ denotes the set $\{i, i+1, \ldots, j\}$. We write Σ^* for the set of finite words over an alphabet Σ, and Σ^ω for the set of infinite words. The length of a finite word w is denoted $|w|$, and if w is infinite, we write $|w| = \infty$. Given a finite or infinite word w and $n \in \mathbb{N}_{>0}$, we denote by $w[n]$ its n-th letter, $w[:n]$ its prefix of length n, $w[n:]$ the suffix obtained by removing its first $n-1$ letters, and more generally $w[n:m]$ for its factor starting at position n up to position m. If $(i_j)_{j\in J}$ is a sequence of indices, we write $w[(i_j)_{j\in J}]$ for the sequence $(w[i_j])_{j\in J}$. In particular, $w[n, m]$ is the pair $(w[n], w[m])$.

2.2 Description of the setting

A finite digraph is a tuple $G = (V, E)$ where V is a finite set of vertices and $E \subseteq V \times V$ is a finite set of (directed) edges. An edge-labelling function is a function $\lambda : E \to \{ ✔, ━, ✘ \}$. Label ✔ (resp. ━, ✘) stands for a good (resp. neutral, bad) edge. Paths are (finite or infinite) sequences of consecutive edges (that is, elements of $E^* \cup E^\omega$).

A finite digraph $G = (V, E)$ with an edge-labelling $\lambda : E \to \{ ✔, ━, ✘ \}$ and a distinguished initial vertex $v_{\text{init}} \in V$ (we write $\langle G, \lambda, v_{\text{init}} \rangle$) defines a *population game* $\mathsf{Pop}(\langle G, \lambda, v_{\text{init}} \rangle)$ as follows. A *move* is an infinite path $\pi \in E^\omega$; we say it is *initial* when it starts at v_{init}. Later we will abusively use the terminology move for finite portions of real moves, when it is clear in the context. A *population strategy* σ is an infinite sequence of initial moves $(\pi_i)_{i\in\mathbb{N}_{>0}}$. It is said *winning for population size N* ($N \in \mathbb{N}_{>0}$) whenever there exists $i(N) \in \mathbb{N}_{>0}$ such that $\lambda(\pi_{i(N)}[N]) =$ ✔ and for all $j < i(N)$, $\lambda(\pi_j[N]) =$ ━. It is said *(uniformly) winning* whenever it is winning for all population sizes. The "uniformly" adverb insists on the fact that the population strategy is independent of the population size: a strategy is not a sequence of paths of length N for each N, but a single family of infinite paths.

Later, we will also use the more permissive concept of *non-losing* population strategy $\sigma = (\pi_i)_{i\in\mathbb{N}}$ such that for every $N \in \mathbb{N}_{>0}$, if there exists an integer $i(N)$ such that $\lambda(\pi_{i(N)}[N]) =$ ✗, then there exists $j < i(N)$, $\lambda(\pi_j[N]) =$ ✓. Note that a winning strategy is non-losing; however a non-losing strategy might be non winning, if for some $N \in \mathbb{N}_{>0}$, for every i, $\lambda(\pi_i[N]) =$ −.

We are now in a position to formally state our decision problem:

WinPop
Input: A finite digraph $G = (V, E)$ with an edge-labelling $\lambda : E \to \{$ ✓, −, ✗ $\}$ and a distinguished initial vertex $v_{\text{init}} \in V$.
Question: Does there exist a winning population strategy in $\mathsf{Pop}(\langle G, \lambda, v_{\text{init}}\rangle)$?

Our main contribution is to establish that the above decision problem is PSPACE-complete.

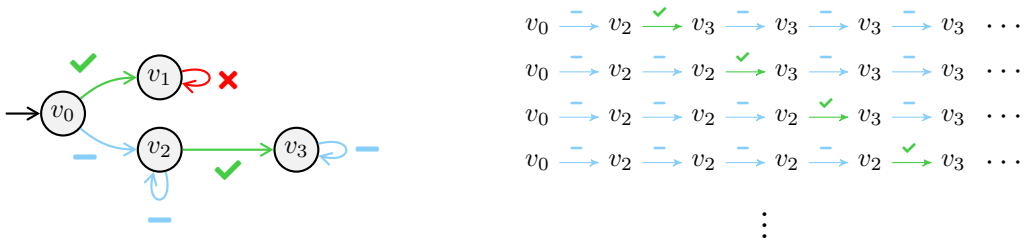
3 Playing graph-based population games

3.1 Examples

We provide a couple of examples to illustrate graph-based population games,

▶ **Example 2.** Let us examine the labelled graph in Figure 1a. A solution is depicted on Figure 1b. It first selects the path that loops 0 times on the initial vertex, then the path that loops once, then twice, and so on. This permits to win for every population size one by one in increasing order.

▶ **Example 3.** We give in Figure 3 a negative instance of the problem. The top branch of the graph is the only one that can make a population of size one win, but it makes all other sizes of population lose. All sizes of population can win by using the lower branch and looping enough, but at no time is it possible to have already won for all population sizes above 1. Hence it is never possible to use the first branch to make size one win.



(a) Labelled graph. **(b)** A non-losing strategy.

Figure 3 Example of a labelled graph $\langle G, \lambda, v_{\text{init}}\rangle$ (left) for which there is no winning population strategy in $\mathsf{Pop}(\langle G, \lambda, v_{\text{init}}\rangle)$; a non-losing strategy is given on the right.

▶ **Example 4.** We now consider the example in Figure 4, with parameters $n_1, n_2 \in \mathbb{N}_{>0}$. The upper branch permits to win for every population size $k+1$ with $k \equiv 0 \mod n_1$. The lower branch permits to win for a number of players that is n_2 less than a population size that has already won. Combining those two types of moves, one can win for every N of the form $1 + \alpha_1 n_1 - \alpha_2 n_2$ with $\alpha_1, \alpha_2 \in \mathbb{N}$. This covers all positive positions if and only if n_1 and n_2 are co-primes, by Bézout's theorem.

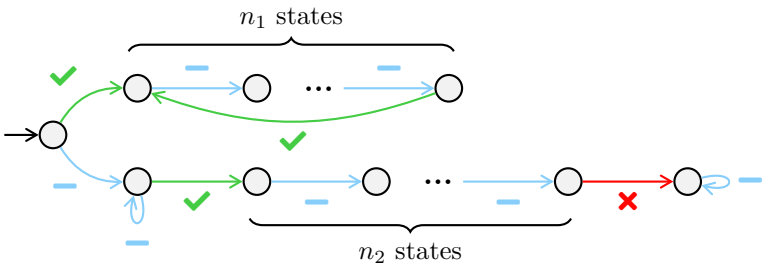
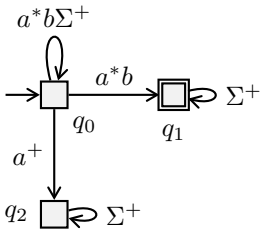


Figure 4 Example of a labelled graph which is a positive instance of WinPop if and only if n_1 and n_2 are co-primes.

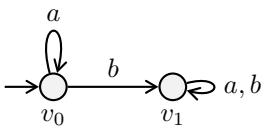
3.2 Relationship with concurrent parameterized games

We mentioned in the introduction the relation between graph-based population games and concurrent parameterized games. The problem WinPop was actually inspired by those concurrent games, so we explain what they are and the relationship.

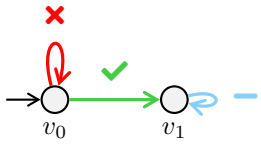
This model of concurrent parameterized games has been first investigated in [4, 5]. A concurrent parameterized game is played on a (finite) graph arena. There is a starting position and a winning condition. Edges of the graph are labelled by regular languages, see Figure 5a for an example. A word w of length N represents a joint move of N players (representing one action per player: player i's action is letter i of w).



(a) An example arena with a reachability objective (target state q_1), where there is a coalition winning strategy.



(b) Deterministic finite automaton that discriminates the three languages labelling the transitions out of q_0: words ending by the transition $v_0 \xrightarrow{a} v_0$ (resp. $v_0 \xrightarrow{b} v_1$, $v_1 \xrightarrow{a,b} v_1$) lead to q_2 (resp. q_1, q_0).



(c) Labelled graph defining the population game corresponding to the parameterized game of Figure 5a. Labels ✔, − and ✖ are obtained via the explanations given in Figure 5b.

Figure 5 Link with concurrent parameterized games [5] with a reachability objective.

Such a game is played as follows: the number of players is finite but arbitrary, and fixed but unknown at the start of the game. In the example depicted on Figure 5a, the objective is to reach the target state (q_1 here) from the initial state q_0, whatever the number of players. An initial move is an infinite word, for instance a^2ba^ω, where the prefix of length N represents the move of the players in case there are N of them. With that move, from q_0, if there are three players, the game will proceed to the target (since $a^2b \in a^*b\Sigma^+$); if there are two players, the game will proceed to q_2 (since $a^2 \in a^+$). In general, a *coalition strategy* associates with every non-empty sequence of vertices an initial move for the last vertex of the sequence, and such a strategy is winning whenever for every $N \in \mathbb{N}_{>0}$, the path obtained by reading from q_0 the prefixes of length N of all moves given by the strategy visits q_1. On the example above, a coalition strategy reduces to an infinite sequence of moves (since q_1 and q_2 are sinks, hence only moves of paths q_0^k for $k \in \mathbb{N}_{>0}$ are meaningful). In the example, there exists a winning coalition strategy, which consists in playing ba^ω from the initial vertex, then

aba^ω, then a^2ba^ω, etc. Under this strategy, if there are N players, the game will loop $(N-1)$ times in q_0 and then proceed to the target q_1. In the context of concurrent parameterized games, the problem is called the coalition problem, and the *reachability coalition problem* for a reachability objective like in the example.

The paramaterized game described in Figure 5a can be turned to a population game as illustrated on Figure 5c: there is a coalition strategy in the game given in Figure 5a if and only if there is a population strategy in the game given in Figure 5c.

First we construct a deterministic finite automaton, which allows to recognize all the words that lead from q_0 to each other state (Figure 5b); then this automaton is turned into a labelled finite graph as on Figure 5c. This construction is possible for all concurrent games with a main state (q_0), and winning/losing sink states (q_1 and q_2).

While concurrent parameterized games with safety objectives have been studied and solved in [5], the techniques which are used are specific to safety objectives, and they do not extend to reachability objectives. The current paper answers the question for arenas with the shape of Figure 5a (one central neutral state, one losing state and one winning state), and the decidability status of the general reachability coalition problem remains open.

4 Deciding how populations uniformly win

We fix for the rest of this section a population game $\mathsf{Pop}(\langle G, \lambda, v_{\text{init}}\rangle)$, where $G = (V, E)$. A population strategy is an infinite sequence of initial moves, each such move being an infinite path in G starting in the initial vertex. To exhibit a winning strategy for positive instances of WinPop, one needs to be able to express the full sequence of paths with a finite description, which we do by identifying some repeating pattern. To do so, we change our point of view: instead of a sequence of infinite paths in the graph, we see strategies as infinite words of sequences of edges. If we represent strategies as grids as in Figure 1b, this means that we switch our vision of the strategy from an infinite sequence of rows to an infinite sequence of columns. We will prove that for positive instances, there always exists a winning strategy that can be decomposed as in Figure 6, with a prefix f followed by iterations of a factor g.

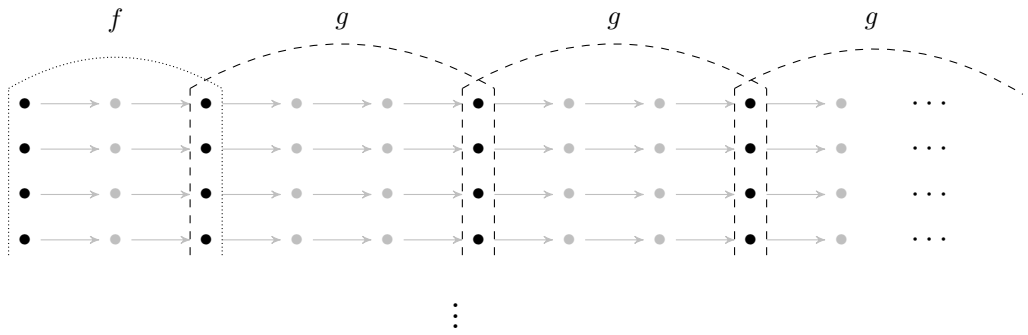


Figure 6 Schematic representation of a winning population strategy decomposition.

More precisely we determine the existence of winning strategies using a finite semigroup, in which we map sequences of edges. A vertical chunk is abstracted by the sequence of endpoints of its edges. To keep the semigroup finite, repetitions are removed. In other words, we only keep the set of pairs of vertices that appear as endpoints of some element in the chunk, in order of appearance. f and g in the above figure yield one element each of the semigroup. The internal operation of the semigroup abstracts the concatenation operation on vertical chunks.

We then characterise labelled graphs for which winning population strategies exist using this semigroup, as announced in Theorem 1. A key element of the proof of the left-to-right implication is to apply Ramsey's theorem to an arbitrary winning population strategy in order to abstract it as a finite prefix followed by infinitely many repetitions of a pattern.

4.1 Frontiers as summaries of strategies

We define frontiers, a tool that allows to summarize parts of the 2D infinite grid representing a strategy as on Figure 1b. Since there are only finitely many frontiers, we will then derive the existence of a frontier that ultimately repeats forever.

▶ **Definition 5.** *A* frontier *is a sequence of pairs of vertices in V of the form $(x_1, y_1), \dots, (x_p, y_p)$ without repetitions, i.e., for all $i < j$, $(x_i, y_i) \neq (x_j, y_j)$. The set of frontiers is denoted $\mathcal{F}$.*

For readability, as a frontier is meant to be a summary of columns of a table as in Figure 1b, we will often write such a sequence of pairs vertically: $\begin{bmatrix} x_1, y_1 \\ \vdots \\ x_p, y_p \end{bmatrix}$. We will do so more generally for sequences of tuples of vertices. Note that frontiers need not abstract any actual sequence of paths.

For any (finite or infinite) sequence $\sigma = s_1, s_2, \cdots$ of elements from a finite set S, we write $\langle \sigma \rangle$ for the finite sequence obtained from σ by removing duplicates, while keeping the order of first appearance. Observe that for any sequence of pairs of vertices $(x_1, y_1), (x_2, y_2), \cdots$, its *reduction* $\langle (x_1, y_1), (x_2, y_2), \cdots \rangle$ is a frontier.

▶ **Definition 6.** *Given three frontiers f, g and h, we write $f \star g \to h$ if there exist triples of vertices $(x_1, y_1, z_1), \dots, (x_p, y_p, z_p) \in V^3$ such that*

$$f = \left\langle \begin{bmatrix} x_1, y_1 \\ \vdots \\ x_p, y_p \end{bmatrix} \right\rangle; \qquad g = \left\langle \begin{bmatrix} y_1, z_1 \\ \vdots \\ y_p, z_p \end{bmatrix} \right\rangle; \qquad \text{and } h = \left\langle \begin{bmatrix} x_1, z_1 \\ \vdots \\ x_p, z_p \end{bmatrix} \right\rangle.$$

A frontier can thus be used to summarize the effect of a sequence of finite portions of moves in G by the reduced sequence of pairs of both their ends. Intuitively, the combination of frontiers with $\star$ corresponds to paths concatenation. Note however that $\star$ is not a function. For instance, with $f = (x_1, y_1), (x_2, y_1), (x_3, y_1)$ and $g = (y_1, z_1), (y_1, z_2)$ one has $f \star g \to (x_1, z_1), (x_2, z_1), (x_3, z_2)$ and $f \star g \to (x_1, z_1), (x_2, z_2), (x_3, z_2)$. One way to turn it into a function, is to lift the operator $\star$ to sets of frontiers: for $F, G \subseteq \mathcal{F}$, $F \star G = \{h \mid \exists f \in F,\ g \in G :\ f \star g \to h\}$. As we will see, this operation yields a semigroup over $2^{\mathcal{F}}$.

We start with a technical lemma expressing that if two sequences of tuples $\mathbf{x}$ and $\mathbf{y}$ are abstracted by frontiers f and g, and $f \star g \to h$, then there is a sequence of tuples that unifies $\mathbf{x}$ and $\mathbf{y}$ and whose abstraction is h. Recall all notations of section 2.1 and in particular, that for a sequence τ, $\tau[i, j]$ denotes the sequence composed of the two elements $\tau[i], \tau[j]$.

▶ **Lemma 7.** *Let $f, g, h \in \mathcal{F}$ be frontiers such that $f \star g \to h$. Suppose we have sequences of tuples of vertices $\mathbf{x} = \begin{bmatrix} \tau_1^{\mathbf{x}} \\ \vdots \\ \tau_r^{\mathbf{x}} \end{bmatrix} \in (V^m)^r$ and $\mathbf{y} = \begin{bmatrix} \tau_1^{\mathbf{y}} \\ \vdots \\ \tau_s^{\mathbf{y}} \end{bmatrix} \in (V^n)^s$ (for some m, n, r, s) such that:*

$$f = \left\langle \begin{bmatrix} \tau_1^{\mathbf{x}}[1, m] \\ \vdots \\ \tau_r^{\mathbf{x}}[1, m] \end{bmatrix} \right\rangle \qquad \text{and} \qquad g = \left\langle \begin{bmatrix} \tau_1^{\mathbf{y}}[1, n] \\ \vdots \\ \tau_s^{\mathbf{y}}[1, n] \end{bmatrix} \right\rangle .$$

Then there exists a sequence of tuples $\mathbf{z} = \begin{bmatrix} \tau_1^{\mathbf{z}} \\ \vdots \\ \tau_p^{\mathbf{z}} \end{bmatrix} \in \left(V^{m+n-1}\right)^p$ *(for some* p*) such that:*

$$\langle \mathbf{x} \rangle = \left\langle \begin{bmatrix} \tau_1^{\mathbf{z}}[:m] \\ \vdots \\ \tau_p^{\mathbf{z}}[:m] \end{bmatrix} \right\rangle \qquad \langle \mathbf{y} \rangle = \left\langle \begin{bmatrix} \tau_1^{\mathbf{z}}[m:] \\ \vdots \\ \tau_p^{\mathbf{z}}[m:] \end{bmatrix} \right\rangle \qquad \text{and} \qquad h = \left\langle \begin{bmatrix} \tau_1^{\mathbf{z}}[1, m+n-1] \\ \vdots \\ \tau_p^{\mathbf{z}}[1, m+n-1] \end{bmatrix} \right\rangle$$

We can now show that the operation $\star$ is associative on $2^{\mathcal{F}}$, and thus that $(2^{\mathcal{F}}, \star)$ is a semigroup (it is even a monoid).

▶ **Lemma 8.** *$(2^{\mathcal{F}}, \star)$ is a semigroup.*

As the $\star$ operation is associative, the notation $F_1 \star \cdots \star F_k$ is well-defined. An element of this product can be decomposed as follows.

▶ **Lemma 9.** *For all $F_1, \ldots, F_n \in 2^{\mathcal{F}}$ and $h \in \mathcal{F}$, we have $h \in F_1 \star \cdots \star F_n$ if and only if there exist $f_1 \in F_1, \ldots, f_n \in F_n$ and a sequence of tuples $\mathbf{x} = \begin{bmatrix} \tau_1^{\mathbf{x}} \\ \vdots \\ \tau_p^{\mathbf{x}} \end{bmatrix} \in \left(V^{n+1}\right)^p$ such that*

$$h = \left\langle \begin{bmatrix} \tau_1^{\mathbf{x}}[1, n+1] \\ \vdots \\ \tau_p^{\mathbf{x}}[1, n+1] \end{bmatrix} \right\rangle \qquad \text{and for all } 1 \leq i \leq n, \quad f_i = \left\langle \begin{bmatrix} \tau_1^{\mathbf{x}}[i, i+1] \\ \vdots \\ \tau_p^{\mathbf{x}}[i, i+1] \end{bmatrix} \right\rangle.$$

We call $(2^{\mathcal{F}}, \star)$ the *frontier semigroup*. To relate (winning) strategies and frontiers, we use wins words, and we consider a morphism $\psi : \{0,1\}^+ \to 2^{\mathcal{F}}$ whose aim is to exhibit winning strategies. Informally, $\psi(0)$ describes sequences of edges in which a ✔ label appears, with no ✖ before, and $\psi(1)$ describes frontiers corresponding to any sequence of edges.

4.2 Wins words: an alternative view on winning strategies

We propose an alternative view on winning population strategies. For a given sequence of initial moves consistent with a non-losing strategy σ, the set of numbers of players for which it has already won at a given time of the sequence is growing as more and more moves are played. We represent this set of achieved wins by a sequence of words in $\{0,1\}^\omega$: the j-th letter of the i-th word is 1 if the game is won after the i-th move when j players are involved.

▶ **Definition 10.** *A* wins word *is an element of $\{0,1\}^+ \cup \{0,1\}^\omega$. Given two wins words $w, w' \in \{0,1\}^\ell$ with $\ell \in \mathbb{N}_{>0} \cup \{\omega\}$, a vertex $v \in V$ and a finite portion of move $\pi \in E^\ell$ where π starts at v, we write $w \overset{v \cdot \pi}{\rightsquigarrow} w'$ if for all $j > 0$, we have either*

- *$w'[j] = w[j] = 1$, or*
- *$\lambda(\pi[j]) =$ ▬ and $w'[j] = w[j]$, or*
- *$\lambda(\pi[j]) =$ ✔ and $w'[j] = 1$.*

If $v = v_{\text{init}}$ is the initial vertex, we simply write $w \overset{\pi}{\rightsquigarrow} w'$. We write $w \rightsquigarrow w'$ when there is π such that $w \overset{\pi}{\rightsquigarrow} w'$ and $\rightsquigarrow^*$ for the reflexive transitive closure of $\rightsquigarrow$. We define the partial ordering $\preceq$ on wins words as: $w \preceq w'$ if and only if $|w| = |w'|$ and $w[j] \leq w'[j]$ for all j.

Note the three constraints on evolutions of wins words in the above definition, that encode the evolution of the winning status for a fixed number of players along a winning play. Population games can be phrased with wins words: one starts from 0^ω and the goal is to read infinite paths in G such that one never sees label ✖ on a position marked 0, and makes sure that every position of the wins word is eventually turned to 1 with a ✔ label.

▶ **Example 11.** Back to Example 2 depicted on Figure 1b. Writing $(\pi_i)_{i\in\mathbb{N}_{>0}}$ for the mentioned winning population strategy, its corresponding sequence of wins words, starting at 0^ω is:

$$0^\omega \overset{\pi_1}{\rightsquigarrow} 10^\omega \overset{\pi_2}{\rightsquigarrow} 110^\omega \overset{\pi_3}{\rightsquigarrow} 1110^\omega \overset{\pi_4}{\rightsquigarrow} \cdots$$

From the above definitions, we immediately derive important properties of wins words:

▶ **Lemma 12.**

- *(Winning strategies as wins words) A population strategy $\sigma = (\pi_i)_{i\in\mathbb{N}_{>0}}$ is winning if and only if the sequence $w_0 \overset{\pi_1}{\rightsquigarrow} w_1 \overset{\pi_2}{\rightsquigarrow} \cdots$ with $w_0 = 0^\omega$ is well-defined (that is, the strategy is non-losing) and the sequence $(w_i)_{i\in\mathbb{N}}$ converges to 1^ω.*
- *(Monotonicity) For all $w \overset{v\cdot\pi}{\rightsquigarrow} w'$, we have $w \preceq w'$. Furthermore, for all $\overline{w}$ such that $w \preceq \overline{w}$, we have $\overline{w} \overset{v\cdot\pi}{\rightsquigarrow} \overline{w'}$ for some $\overline{w'}$ such that $w' \preceq \overline{w'}$.*
- *(Composition) If w is finite, $w \overset{v\cdot\pi}{\rightsquigarrow} w'$ and $\overline{w} \overset{\overline{v}\cdot\overline{\pi}}{\rightsquigarrow} \overline{w'}$ with π a path from v to $\overline{v}$, then $w\overline{w} \overset{v\cdot\pi\overline{\pi}}{\rightsquigarrow} w'\overline{w'}$.*
- *(Invariance under repetitions) If $w \overset{v\cdot\pi}{\rightsquigarrow} w'$ then for all w'' with $w' \preceq w''$ we have $w'' \overset{v\cdot\pi}{\rightsquigarrow} w''$.*

▶ **Definition 13.** *Define the morphism $\psi : \{0,1\}^+ \to 2^{\mathcal{F}}$ as*

$$\psi(0) = \left\{ \begin{bmatrix} x_1, y_1 \\ \vdots \\ x_p, y_p \end{bmatrix} \in \mathcal{F} \mid (\forall i, e_i \overset{def}{=} (x_i, y_i) \in E) \wedge (\exists j, \lambda(e_j) = \checkmark) \wedge (\forall i < j, \lambda(e_i) = -) \right\}$$

$$\psi(1) = \left\{ \begin{bmatrix} x_1, y_1 \\ \vdots \\ x_p, y_p \end{bmatrix} \in \mathcal{F} \mid \forall i, e_i \overset{def}{=} (x_i, y_i) \in E \right\}$$

▶ **Example 14.** Consider the instance described in Figure 1, and the grid from Figure 1b. The set $\psi(00)$ includes the compressed version of columns 1 and 2 on the one hand, and 2 and 3 on the other hand. With our notations, this translates into:

$$\begin{bmatrix} v_0, v_1 \\ v_0, v_0 \end{bmatrix}, \begin{bmatrix} v_1, v_1 \\ v_0, v_1 \\ v_0, v_0 \end{bmatrix} \in \psi(00).$$

Lemma 15 formalises the application of ψ to an arbitrary word. The idea is that pairs of a frontier are played step by step, and whenever the first p pairs have already been played, the playable pairs are all of the first $p+1$ pairs.

▶ **Lemma 15** (Morphism ψ describes intervals of winning paths)**.** *For all $w \in \{0,1\}^n$ and $h \in \mathcal{F}$, we have $h \in \psi(w)$ if and only if there exist vertices $v_1, \ldots, v_p \in V$ and $v'_1, \ldots, v'_p \in V$, and finite portions of moves $\pi_1, \ldots, \pi_p \in E^n$ such that for every $1 \le i \le p$, π_i is a path from v_i to v'_i and*

$$w \overset{v_1\cdot\pi_1}{\rightsquigarrow} \cdots \overset{v_p\cdot\pi_p}{\rightsquigarrow} 1^n \qquad \text{and} \qquad h = \left\langle \begin{bmatrix} v_1, v'_1 \\ \vdots \\ v_p, v'_p \end{bmatrix} \right\rangle.$$

4.3 Using frontiers to decide WinPop

Now that we have defined a finite semigroup to abstract finite intervals of columns of population strategies, we use it to obtain witnesses of existence of a winning strategy. We show that winning strategies can be summarised by two frontiers, one representing the first few columns, and the other a somewhat periodic pattern in the remaining columns.

We introduce two properties of frontiers below. Initial frontiers represent finite intervals of columns which start at the leftmost column of the grid. Meanwhile, a frontier is ω-iterable if, intuitively, in all intervals of columns it represents, a vertex cannot appear on the rightmost side before it appears on the leftmost side. The motivation for this property is that if we cut a solution to the problem into infinitely many intervals of columns mapping to the same frontier, it cannot be the case that columns to the right of each interval progress faster than columns on the left: we would get an infinite sequence of columns, each one making progress after the next. This is represented in Figure 2: we cannot build a valid infinite grid as every column needs to make progress after the next one.

▶ **Definition 16.** *A frontier is* initial *if all its pairs are in* $\{v_{init}\} \times V$. *A frontier* g *is* ω-iterable *if there exist* $x_1, \dots, x_p, y_1, \dots, y_r, z_1, \dots, z_r \in V$ *such that* $g = \begin{bmatrix} x_1, x_1 \\ \vdots \\ x_p, x_p \\ y_1, z_1 \\ \vdots \\ y_r, z_r \end{bmatrix}$ *with* $z_i \in \{x_1, \dots, x_p, y_1, \dots, y_{i-1}\}$ *for all* i.

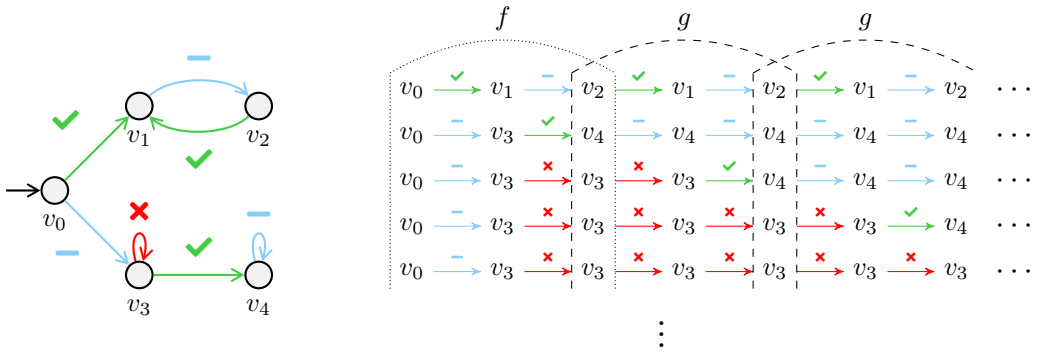
When a sequence of paths in the graph is sliced into intervals of columns, themselves abstracted into frontiers, the first one is initial. An ω-iterable frontier starts with (x_i, x_i)-pairs that can be all concretized by one (infinite) move (by repeating a looping path on x_i). The next ones have to be concretized by iterating families of moves, covering the slices associated with the ω-iterable frontier in increasing order, as formalised in the rest of this section.

▶ **Example 17.** Consider the graph in Figure 7a, and a winning strategy described in Figure 7b. Consider frontiers f and g as described on Figure 7b. The first frontier is $f = \begin{bmatrix} v_0, v_2 \\ v_0, v_4 \\ v_0, v_3 \end{bmatrix}$ and is initial, the second one is $g = \begin{bmatrix} v_2, v_2 \\ v_4, v_4 \\ v_3, v_4 \\ v_3, v_3 \end{bmatrix}$ and is ω-iterable.

We start with the observation that if two frontiers can be composed, then the order of appearance of vertices on their "interface" must be the same.

▶ **Lemma 18.** *Let* $f, g \in \mathcal{F}$ *with* $f = \begin{bmatrix} x_1, y_1 \\ \vdots \\ x_r, y_r \end{bmatrix}$ *and* $g = \begin{bmatrix} x'_1, y'_1 \\ \vdots \\ x'_s, y'_s \end{bmatrix}$. *Assume* $f * g \to h$ *for some* $h \in \mathcal{F}$. *Then we have* $\left\langle \begin{bmatrix} y_1 \\ \vdots \\ y_r \end{bmatrix} \right\rangle = \left\langle \begin{bmatrix} x'_1 \\ \vdots \\ x'_s \end{bmatrix} \right\rangle$.

We can now get to the core of the proof: We will first construct a winning strategy from an initial and an ω-iterable frontier (Theorem 20). Then we will extract such two frontiers from an arbitrary winning strategy (Theorem 22). The two results yield the desired characterization of positive instances of the problem (Theorem 1).



(a) The labelled graph. **(b)** A possible winning population strategy.

Figure 7 A positive instance of WinPop, and frontiers on its winning strategy.

A *slice* of length $n \in \mathbb{N}_{>0} \cup \{\omega\}$ is a finite sequence of pairs $(v_i, \pi_i)_{1 \leq i \leq p} \in (V \times E^n)^p$ such that for every i, π_i starts at v_i. Note that while a slice is always a finite sequence, its length can be infinite, i.e, it can contain infinite words. Given a sequence of distinct vertices $x_1, \dots, x_k$, we say that the slice starts from $x_1, \dots, x_k$ if $\langle v_1, \dots, v_p \rangle = x_1, \dots, x_k$ and that it ends in $x_1, \dots, x_k$ if it has finite length and $\langle v'_i, \dots, v'_p \rangle = x_1, \dots, x_k$ where π_i ends at v'_i. The *result* of the slice is the wins word $w \in \{0,1\}^n$ such that $0^n \overset{v_1 \cdot \pi_1}{\rightsquigarrow} \cdots \overset{v_p \cdot \pi_p}{\rightsquigarrow} w$, if defined.

We will build a winning strategy by constructing slices from frontiers and then patching them together. The patching operation is described in the following lemma.

▶ **Lemma 19.** *Suppose we have a slice* $\mathbf{s}$ *starting from* $x_1, \dots, x_{p(x)}$ *and ending in* $y_1, \dots, y_{p(y)}$ *with result* w *and another one* $\mathbf{s}'$ *starting from* $y_1, \dots, y_{p(y)}$ *with result* w'*. Then we have a slice* $\mathbf{s}''$ *starting from* $x_1, \dots, x_{p(x)}$ *with result* ww'*.*

Furthermore if $\mathbf{s}'$ *is finite and ends in* $z_1, \dots, z_{p(z)}$*, then so does* $\mathbf{s}''$*.*

This allows to state the first implication of the main result.

▶ **Theorem 20** (Frontiers to strategy)**.** *Assume there exist* $f, g \in \psi(0^+)$ *such that:*

- f *is initial,*
- $f * g \to f$*,*
- g *is* ω*-iterable,*
- $g * g \to g$*.*

Then there exists a winning population strategy in $\mathsf{Pop}(\langle G, \lambda, v_{init} \rangle)$*.*

Proof sketch. Let us give the high-level idea of the proof. The full proof can be found in Appendix A. We translate the frontiers f and g into slices using Lemma 15. We show that we can win on every position using the following process: we consider chunks of positions from left to right one by one. On each one, we apply the slice given by g to win on those positions. The initial frontier f lets us complete this slice into a finite sequence of initial finite moves: roughly speaking, we use the slice given by f to extend it to the left and infinitely many iterations of prefixes of the slice of g to extend it to the right (which is possible thanks to ω-iterability). We finally patch those slices together using Lemma 19. ◀

We now show the other implication, i.e., that a winning strategy yields suitable frontiers. The proof will use Ramsey's theorem on infinite graphs, which we recall below.

▶ **Theorem 21** (Ramsey's theorem on infinite graphs [14])**.** *In an infinite complete graph with edges coloured by finitely many colours, there is an infinite clique whose edges all have the same colour.*

This use of Ramsey's theorem is common in the study of the interplay between finite semigroups and infinite words: if we have an infinite sequence of elements of a finite semigroup, we can cut it into a finite prefix and infinitely many factors that all evaluate to the same (idempotent) element in the semigroup. This idea appears already, for instance, in the complementation construction for Büchi automata by the eponymous author [8].

▶ **Theorem 22** (Strategy to frontiers). *If there is a winning population strategy in* $\mathsf{Pop}(\langle G, \lambda, v_{init}\rangle)$*, then there exist* $f, g \in \psi(0^+)$ *such that*

- f *is initial,*
- g *is* ω*-iterable,*
- $f * g \to f$,
- $g * g \to g$.

Proof sketch. This proof is made in two steps :

1. We build an infinite graph with coloured edges, which depends on the winning strategy;
2. We use Ramsey's Theorem on this graph to extract f and g.

Let σ be a winning population strategy, and consider the representation where moves are depicted in successive lines as in Figure 7b. We build a graph whose set of vertices is $\mathbb{N}$, and whose edge between k and ℓ is coloured by a pair (f, g), where:

- f is the frontier summarizing the strategy between the initial column and column k,
- g is the frontier summarizing the strategy between column k and column ℓ.

As the edge-labelled graph has a finite number of vertices, the graph has a finite number of colours, and we can apply Ramsey's theorem on it.

Let (f, g) be the colour of the infinite subgraph given by Ramsey. We note that f is an initial frontier, since all moves given by σ start at v_{init}. By definition of the colour of an edge, the right part of f coincides with the left part of g as well as the right part of g. This allows to infer that $f * g \to f$ and $g * g \to g$. The ω-iterability of g follows from the fact that g is indeed iterated infinitely many times by σ. The full proof can be found in Appendix B. ◀

5 The complexity of WinPop

In this section, we establish the precise complexity of WinPop. While Theorem 1 provides witnesses for the existence of winning strategies, we now show how to look for those witnesses in polynomial space. After, we exhibit a matching complexity lower bound.

▶ **Theorem 23.** *WinPop is PSPACE-complete.*

5.1 Complexity upper bound

To begin with, we observe that deciding if a frontier is in $\psi(0^+)$ comes down to exploring a graph of exponential size.

▶ **Lemma 24.** *Given a frontier* $f \in \mathcal{F}$*, one can check in polynomial space whether* $f \in \psi(0^+)$.

Proof. We build the following graph, of exponential size. Vertices are frontiers, and edges go from each frontier to the ones that can be obtained from it by composition with an element of $\psi(0)$. Formally, $\widehat{G} = (\mathcal{F}, \widehat{E})$, with $(g, h) \in \widehat{E}$ if and only if there exists $g' \in \psi(0)$ such that $g \star g' \to h$. This condition is easily checked in polynomial space.

Clearly $f \in \psi(0^+)$ if and only if there is a path in $\widehat{G}$ from an element of $\psi(0)$ to f. This can be checked by guessing a path in $\widehat{G}$ on the fly. Since one can store a vertex and check the existence of an edge in polynomial space, we get a non-deterministic polynomial space algorithm. We conclude using Savitch's theorem. ◀

▶ **Proposition 25.** *WinPop is in PSPACE.*

Proof. We non-deterministically guess $f, g \in \mathcal{F}$ so that f is initial, g is ω-iterable, $f \star g \to f$ and $g \star g \to g$. Those conditions can easily be checked in PSPACE.

Then, we check that there exist $k, \ell \in \mathbb{N}_{>0}$ such that $f \in \psi(0^k)$ and $g \in \psi(0^\ell)$, i.e., if $f, g \in \psi(0^+)$. This condition can also be checked in polynomial space by Lemma 24. ◀

5.2 Complexity lower bound

We match the PSPACE upper bound with a lower bound, reducing from the termination problem for deterministic Turing machines (DTMs) with unary bounded space. We define DTMs as usual, and simply fix the notations.

We fix a finite alphabet Σ and define a deterministic Turing machine as a tuple $\mathcal{M} = (S_\mathcal{M}, \Sigma, \delta_\mathcal{M}, s_{\text{init}}, F)$, with $\delta_\mathcal{M} : S_\mathcal{M} \times \Sigma \to S_\mathcal{M} \times \Sigma \times \{\leftarrow, \rightarrow\}$ the transition, reading a state and a letter, updating both and then moving left or right on the tape.

A configuration of an n-bounded Turing machine is a word of the form $u(s, a)v$ with $u, v \in \Sigma^*$, $s \in S_\mathcal{M}$ and $a \in \Sigma$, so that $|u| + |v| + 1 = n$. The initial configuration is $(s_{\text{init}}, b)b^{n-1}$.

The input is a deterministic Turing machine along with a number $n \in \mathbb{N}$ in unary. The question is whether the run from the initial configuration eventually reaches a configuration with a state in F. This problem is well-known to be PSPACE-complete.

The intuition of the reduction is the following. We see configurations of the Turing machine as words of fixed length over an alphabet $\Gamma = \Sigma \cup (S_\mathcal{M} \times \Sigma) \cup \{\#\}$, with $\#$ a fresh letter. We then in turn encode those letters as words of the form $0^i 10^{|\Gamma|-i-1}$ over $\{0, 1\}$. Note that all those words have the same length $|\Gamma|$. The sequence of configurations of the run of the DTM can then be seen as a (potentially infinite) word over $\{0, 1\}$, two consecutive configurations being separated by a $\#$. Since the machine is deterministic, for all $i \geq n + 1$, the i-th letter (as a word over Γ) is determined by the $(i - n - 1)$-th, $(i - n)$-th and $(i - n + 1)$-th letters.

We build a graph that computes this sequence by *reading* and *writing*. Reading a letter $0^i 10^{|\Gamma|-i-1}$ means going through a sequence of states with labels $-^i$ ✗ $-^{|\Gamma|-i-1}$. This will lead to a loss if the sequence of bits on these positions is of the form $0^j 10^{|\Gamma|-j-1}$ with $j \neq i$. Writing $0^i 10^{|\Gamma|-i-1}$ means going through a sequence of states with labels $-^i$ ✓ $-^{|\Gamma|-i-1}$. If the sequence of bits was $0^{|\Gamma|}$ before, we will obtain the desired sequence. If it was already $0^i 10^{|\Gamma|-i-1}$, nothing changes. This is where the determinism of the machine is important: the graph will never be able to write two different letters at the same position. Our graph will have a path writing the initial configuration. It will also be able to read three consecutive letters at any positions $i - 1, i, i + 1$, infer the $(i + n + 1)$-st letter, skip $n + 1$ positions and write that letter. Finally, it can read a letter from $\Sigma \times F$, and turns all following positions to 1, as well as the position just before. Hence if we encounter a final state during the computation, this branch will allow us to turn all positions to 1 by applying it sufficiently many times.

The full proof is given in Appendix C.

6 Conclusion and discussion

We have shown PSPACE-completeness of the WinPop problem for so-called *population games*. The problem coincides with a restricted but challenging form of grid tiling problem, which we solve using new algebraic techniques. This problem also coincides with a restricted class of

parameterized concurrent games with reachability objectives. This lets us hope to push the decidability frontier of the two families of problems even further: In particular, we may be able to expand on the techniques developed here to solve the open problem of parameterized concurrent reachability games on arbitrary finite arenas, mentioned as open in [5]. As a first step, one could try to obtain decidability for arenas made of a DAG with loops, by seeing them as several nested arenas with the same shape as the one in Figure 5a. Our result might also be useful to better understand some open tiling problems such as those mentioned in [7].

Beyond deciding the existence of a winning population strategy *for all population sizes*, determining assumptions on the population sizes that guarantee the existence of a winning strategy seems hard. Let us elaborate on that natural extension of WinPop. Given a population game $\mathsf{Pop}(\langle G, \lambda, v_{\text{init}}\rangle)$, one can define its *winning region* as the set of wins words $w \in \{0,1\}^*$ from which there is a winning population strategy, i.e., such that there is a sequence $w_0 \rightsquigarrow w_1 \rightsquigarrow \cdots$ with $w_0 = w$ and such that for all i there exists j such that $w_j[i] = 0$. WinPop corresponds to deciding whether 0^ω belongs to the winning region. In light of Lemma 15, one can wonder if it is possible to extend our construction and PSPACE algorithm to compute the winning region. Indeed, morphism ψ applies to all words in $\{0,1\}^*$. We provide here a partial, somewhat surprising, answer: in general, the winning region is not an ω-regular language (the reader is referred to [16] for definitions and properties of ω-regular languages), as illustrated by the following example:

▶ **Example 26.** Let $W \subseteq \{0,1\}^\omega$ be the winning region for the population game associated with the graph from Figure 8.

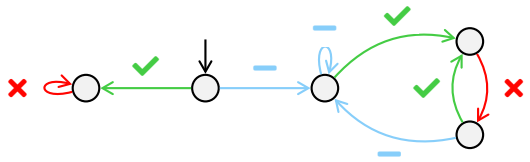


Figure 8 A labelled graph whose associated winning region is not ω-regular.

There are two ways to get new 1s in a word. We can either use the left branch, which turns the first position to 1 but needs all following positions to be 1 already. Or we can use the right branch. It lets us put a 1 on all positions which are followed by a 1. In other words, we can turn the last 0s of all blocks of 0 to 1s. Suppose we start with a configuration in $0(10^+)^\omega$. Then there is a winning strategy if and only if we can eliminate all 0s (except the first one) by applying the right branch finitely many times. This is the case if and only if the blocks of 0 are uniformly bounded.

We obtain that $W \cap 0(10^+)^\omega$ is the set of words starting with 0, with infinitely many ones and uniformly bounded blocks of 0, which is not an ω-regular language.

References

1 Jan-Henrik Altenbernd, Wolfgang Thomas, and Stefan Wöhrle. Tiling systems over infinite pictures and their acceptance conditions. In *Proc. of the International Conference on Developments in Language Theory (DLT'02)*, volume 2450 of *LNCS*, pages 297–306. Springer, 2002. doi:10.1007/3-540-45005-X_26.

2 Robert Berger. *The undecidability of the domino problem*, volume 66. American Mathematical Society Providence, 1966.

3 Nathalie Bertrand, Patricia Bouyer, Luc Lapointe, and Corto Mascle. Reach together: How populations win repeated games, 2025. doi:10.48550/arXiv.2510.02984.

4 Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Concurrent parameterized games. In *Proc. of the 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'19)*, volume 150 of *LIPIcs*, pages 31:1–31:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.31.

5 Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Synthesizing safe coalition strategies. In *Proc. of the 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'20)*, volume 182 of *LIPIcs*, pages 39:1–39:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.39.

6 Nathalie Bertrand, Miheer Dewaskar, Blaise Genest, Hugo Gimbert, and Adwait Amit Godbole. Controlling a population. *Logical Methods in Computer Science*, 15(3), 2019. doi:10.23638/LMCS-15(3:6)2019.

7 Achim Blumensath, Olivier Carton, and Thomas Colcombet. Asymptotic monadic second-order logic. In *Proc. of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS'14)*, volume 8634 of *LNCS*, pages 87–98. Springer, 2014. doi:10.1007/978-3-662-44522-8_8.

8 J. Richard Büchi. On a decision method in restricted second order arithmetic. *Proceedings of the Internatational Congress on Logic, Methodology and Philosophy of Science*, 1960.

9 Thomas Colcombet, Nathanaël Fijalkow, and Pierre Ohlmann. Controlling a random population. *Logical Methods in Computer Science*, 17(4), 2021. doi:10.46298/LMCS-17(4:12)2021.

10 Olivier Finkel. On recognizable languages of infinite pictures. *International Journal of Foundations of Computer Science*, 15(06):823–840, 2004. doi:10.1142/S0129054104002777.

11 Hugo Gimbert, Corto Mascle, and Patrick Totzke. Optimally controlling a random population. Technical report, CoRR, 2025. arXiv:2411.15181.

12 Emile Hazard and Denis Kuperberg. Explorable automata. In *Proc. of the 31st EACSL Annual Conference on Computer Science Logic (CSL'23)*, volume 252 of *LIPIcs*, pages 24:1–24:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.24.

13 Oliver Matz and Nicole Schweikardt. Expressive power of monadic logics on words, trees, pictures, and graphs. *Logic and Automata*, 2:531–552, 2008.

14 F. P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, s2-30(1):264–286, 1930. doi:10.1112/plms/s2-30.1.264.

15 Taylor J Smith and Kai Salomaa. Concatenation operations and restricted variants of two-dimensional automata. In *Proc. of the 47th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'21)*, volume 12607 of *LNCS*, pages 147–158. Springer, 2021. doi:10.1007/978-3-030-67731-2_11.

16 Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 133–191. Elsevier and MIT Press, 1990. doi:10.1016/B978-0-444-88074-1.50009-3.

A Proof of Theorem 20

▶ **Theorem 20** (Frontiers to strategy). *Assume there exist* $f, g \in \psi(0^+)$ *such that:*

- f *is initial,*
- g *is* ω*-iterable,*
- $f * g \to f$,
- $g * g \to g$.

Then there exists a winning population strategy in $\mathsf{Pop}(\langle G, \lambda, v_{init}\rangle)$.

Proof. Since $f \in \psi(0^k)$ and $g \in \psi(0^\ell)$ for some $k, \ell \in \mathbb{N}_{>0}$, by Lemma 15 we have slices $(v_i^f, \pi_i^f)_{1\le i\le m} \in \left(V \times E^k\right)^m$ and $(v_i^g, \pi_i^g)_{1\le i\le n} \in (V \times E^\ell)^n$ with result respectively 1^k and 1^ℓ. We note $v_i'^f$ and $v_i'^g$ respectively the endpoints of π_i^f and π_i^g respectively, for every i.

By Lemma 18, since $f \star g \to f$ and $g \star g \to g$, the sequences $\left\langle \begin{bmatrix} v_1'^f \\ \vdots \\ v_m'^f \end{bmatrix} \right\rangle$, $\left\langle \begin{bmatrix} v_1^g \\ \vdots \\ v_n^g \end{bmatrix} \right\rangle$ and $\left\langle \begin{bmatrix} v_1'^g \\ \vdots \\ v_n'^g \end{bmatrix} \right\rangle$ are equal. Let $\mathbf{v}$ be that sequence.

Since f is initial, all v_i^f are v_{init}. As a consequence, we have a slice $\mathbf{s}_f$ from $\lceil v_{\text{init}} \rceil$ to $\mathbf{v}$. Since $(v_i^g, \pi_i^g)_{1 \le i \le n} \in \left(V \times E^\ell\right)^n$ is a slice from $\mathbf{v}$ to itself with result 1^ℓ, by applying Lemma 19 we can obtain, for every $N \in \mathbb{N}_{>0}$, a slice $\mathbf{s}_N$ from $\lceil v_{\text{init}} \rceil$ to $\mathbf{v}$ with result $1^{k+\ell \cdot N}$.

Since g is ω-iterable, it is of the form $\begin{bmatrix} x_1, x_1 \\ \vdots \\ x_p, x_p \\ y_1, z_1 \\ \vdots \\ y_r, z_r \end{bmatrix}$ with $z_j \in \{x_1, \dots, x_p, y_i, \dots, y_{j-1}\}$ for all j. For each j, let $m(j)$ be the minimal index such that $v_{m(j)}^g = y_j$. The slice $(v_i^g, \pi_i^g)_{1 \le i < m(j)}$ starts from $\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ y_1 \\ \vdots \\ y_j \end{bmatrix} \right\rangle$ and ends in $\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ y_1 \\ \vdots \\ y_{j'} \end{bmatrix} \right\rangle$ with $j' < j$. All these slices have well-defined results, as prefixes of $(v_i^g, \pi_i^g)_{1 \le i \le n}$, which has a result.

By Lemma 19, we can thus compose such slices to obtain a slice from $\mathbf{v}$ to $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$ with some result w. For all $N \in \mathbb{N}$, we can apply the lemma again with $\mathbf{s}_N$ to get a slice $\mathbf{s}'_N$ from $\lceil v_{\text{init}} \rceil$ to $\begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$ with result $1^{k+\ell \cdot N} w$.

Now consider the minimal index m_x such that $(v_{m_x}, v'_{m_x}) = (x_p, x_p)$. The slice $(v_i^g, \pi_i^g)_{1 \le i \le m_x}$ has a result w_x, and it can be iterated to form an infinite slice: Let $\mathbf{s}_\omega = (v_i^g, (\pi_i^g)^\omega)_{1 \le i \le m_x}$. It is an infinite slice from $\mathbf{x}$ with result w_x^ω. For all $N \in \mathbb{N}$, we can apply Lemma 19 with $\mathbf{s}'_N$ and $\mathbf{s}_\omega$ to obtain an infinite slice $(v_{\text{init}}, \varpi_{N,i})_{1 \le i \le m_N}$ from $\lceil v_{\text{init}} \rceil$ with result $1^{k+\ell \cdot N} w w_x^\omega$.

As a consequence, for all $N \in \mathbb{N}$ we have a sequence of moves $0^\omega \overset{u_{N,1}}{\rightsquigarrow} \cdots \overset{u_{N,m_N}}{\rightsquigarrow} 1^{k+\ell \cdot N} w w_x^\omega$. We can apply this for every $N \in \mathbb{N}$ consecutively to obtain a winning strategy: by monotonicity (see related item in Lemma 12), after applying the N-th sequence, the wins word w_N obtained is well-defined and $1^{k+\ell \cdot N} 0^\omega \preceq w_N$. By Lemma 12 (Winning strategies as wins words), we have a winning strategy. ◀

B Proof of Theorem 22

▶ **Theorem 22** (Strategy to frontiers). *If there is a winning population strategy in* $\mathsf{Pop}(\langle G, \lambda, v_{init} \rangle)$*, then there exist* $f, g \in \psi(0^+)$ *such that*

- f *is initial,*
- g *is* ω*-iterable,*
- $f * g \to f$*,*
- $g * g \to g$*.*

Proof. Assume there is a population winning strategy $\sigma = (\pi_i)_{i \in \mathbb{N}_{>0}}$ in $\mathsf{Pop}(\langle G, \lambda, v_{\text{init}} \rangle)$. Define $v_{i,j}$ the vertex of the graph G after finite move $\pi_i[:j]$ (from v_{init}). We obtain a grid of vertices as in Figure 7b.

Consider now the infinite complete undirected graph $\widehat{G}$ with coloured edges, defined as follows:

- Its vertices are $\mathbb{N}$;
- Let $k < \ell \in \mathbb{N}$, the edge $\{k, \ell\}$ is coloured by the pair of frontiers $(f_k, g_{k,\ell})$ where

$$f_k = \left\langle \begin{bmatrix} v_{0,0}, v_{0,k} \\ v_{1,0}, v_{1,k} \\ \vdots \end{bmatrix} \right\rangle \qquad \text{and} \qquad g_{k,\ell} = \left\langle \begin{bmatrix} v_{0,k}, v_{0,\ell} \\ v_{1,k}, v_{1,\ell} \\ \vdots \end{bmatrix} \right\rangle.$$

  Note that f_k is always initial. Informally, when moves are written in lines as in Figure 7b, f_k is the frontier obtained from columns between 0 and k, and $g_{k,\ell}$ is the frontier obtained from columns between k and ℓ.

We apply Ramsey's theorem (recalled as Theorem 21) to $\widehat{G}$. Let $S \subseteq \mathbb{N}$ such that the complete sub-graph induced by S has only one colour (f, g). By construction, f is initial. To show that $g * g \to g$, consider $k < \ell < m \in S$, and the sequence of triples $\begin{bmatrix} v_{0,k}, v_{0,\ell}, v_{0,m} \\ v_{1,k}, v_{1,\ell}, v_{1,m} \\ \vdots \end{bmatrix}$. By construction:

$$\left\langle \begin{bmatrix} v_{0,k}, v_{0,\ell} \\ v_{1,k}, v_{1,\ell} \\ \vdots \end{bmatrix} \right\rangle = \left\langle \begin{bmatrix} v_{0,\ell}, v_{0,m} \\ v_{1,\ell}, v_{1,m} \\ \vdots \end{bmatrix} \right\rangle = \left\langle \begin{bmatrix} v_{0,k}, v_{0,m} \\ v_{1,k}, v_{1,m} \\ \vdots \end{bmatrix} \right\rangle = g$$

It therefore witnesses the fact that $g * g \to g$. A similar proof shows that $f * g \to f$, by considering columns 0, i and j.

Now we show that $g \stackrel{\text{def}}{=} \begin{bmatrix} x_1, y_1 \\ \vdots \\ x_n, y_n \end{bmatrix}$ is ω-iterable. Let $s : \mathbb{N} \to S$ be an enumeration of S, i.e., an increasing function such that $S = s(\mathbb{N})$. We proceed in two steps:

- First we show that for all $i \in [\![1, n]\!]$, $y_i \in \{x_1, \dots, x_i\}$. Let $i \in [\![1, n]\!]$, and let m be the minimal index such that there exist $k < \ell \in S$ such that $(v_{m,k}, v_{m,\ell}) = (x_i, y_i)$. Such an m exists by definition of g. Let $p \in S$ such that $p > \ell$. By minimality of m, we must have $(v_{j,\ell}, v_{j,p}) \in \{(x_1, y_1), \dots, (x_{i-1}, y_{i-1})\}$ for all $j < m$. By definition of S, we have $\left\langle \begin{bmatrix} v_{0,\ell}, v_{0,p} \\ v_{1,\ell}, v_{1,p} \\ \vdots \end{bmatrix} \right\rangle = g$, thus $(v_{m,\ell}, v_{m,p}) \in \{(x_1, y_1), \dots, (x_i, y_i)\}$. In particular we have $y_i \in \{x_1, \dots, x_i\}$.
- Now we assume by contradiction that g is not ω-iterable. Since $y_i \in \{x_1, \dots, x_i\}$ for all i, this means that there exist $j < k$ such that:
  - $x_i = y_i$ for all $i < j$,
  - $x_j \neq y_j$ (hence $y_j \in \{x_1, \dots, x_{j-1}\}$),
  - $x_k = y_k$ with $y_k \notin \{x_1, \dots, x_{k-1}\}$.

For all $\ell \in \mathbb{N}$ let $\alpha(\ell)$ be the minimal index such that $(v_{\alpha(\ell),s(\ell)}, v_{\alpha(\ell),s(\ell+1)}) = (x_j, y_j)$. We show that $(\alpha(\ell))_\ell$ is an increasing sequence. Fix ℓ: $(v_{\alpha(\ell),s(\ell)}, v_{\alpha(\ell),s(\ell+1)}) = (x_j, y_j)$ with $y_j \in \{x_1, \dots, x_{j-1}\}$, and for every $i < \alpha(\ell)$, $v_{i,s(\ell)} = v_{i,s(\ell+1)} \in \{x_1, \dots, x_{j-1}\}$. This implies that $\alpha(\ell+1) > \alpha(\ell)$.

By assumption, there is m such that $(v_{m,s(0)}, v_{m,s(1)}) = (x_k, x_k)$. Since $x_k \notin \{x_1, \dots, x_{k-1}\}$ and $g = \left\langle \begin{bmatrix} v_{0,s(\ell)}, v_{0,s(\ell+1)} \\ v_{1,s(\ell)}, v_{1,s(\ell+1)} \\ \vdots \end{bmatrix} \right\rangle$ for every ℓ, we deduce $v_{m,s(\ell)} = x_k$ for every ℓ.

Let ℓ be such that $\alpha(\ell) > m$ (it exists since the sequence $(\alpha(\ell))_\ell$ is increasing). Then $(v_{m,s(\ell)}, v_{m,s(\ell+1)}) = (x_k, x_k)$ with $x_k \notin \{x_1, \dots, x_{k-1}\}$. On the other hand, by definition of $\alpha(\ell)$, $(v_{\alpha(\ell),s(\ell)}, v_{\alpha(\ell),s(\ell+1)}) = (x_j, y_j)$ and for every $i < \alpha(\ell)$, $(v_{i,s(\ell)}, v_{i,s(\ell+1)}) \in \{(x_1, y_1), \dots, (x_{j-1}, y_{j-1})\}$. This yields a contradiction since $j < k$.

We conclude that g is ω-iterable.

Lastly, we prove that f and g are in $\psi(0^+)$. Let $k \in S$, by definition we have $f = f_k = \left\langle \begin{bmatrix} v_{0,0}, v'_{0,0} \\ v_{1,0}, v'_{1,0} \\ \vdots \end{bmatrix} \right\rangle$. where move $\pi_i[:k]$ ends at $v'_{i,0}$ for every i. Since σ is a winning strategy, there is an index i such that $0^n \overset{v_{0,0}\cdot\pi_0[:k]}{\rightsquigarrow} \dots \overset{v_{i,0}\cdot\pi_i[:k]}{\rightsquigarrow} 1^n$. By Lemma 15, we obtain that $f \in \psi(0^+)$. The proof for g is similar. ◀

C Complexity lower-bound

▶ **Proposition 27.** *WinPop is PSPACE-hard.*

Proof. Let $\mathcal{M} = (S_\mathcal{M}, \Sigma, \delta_\mathcal{M}, s_{\text{init}}, F)$ be a DTM and let $n \in \mathbb{N}$.

Let $\Gamma = \Sigma \cup (S_\mathcal{M} \times \Sigma) \cup \{\#\}$, and let $m = |\Gamma|$. Let β be a bijection between Γ and $[\![0, m-1]\!]$. Define, for all $\gamma \in \Gamma$, $\phi(\gamma) = 0^{\beta(\gamma)}10^{m-1-\beta(\gamma)}$. Since $\mathcal{M}$ is deterministic, it has a single run from $(s_{\text{init}}, b)b^{n-1}$. Let $c_0, c_1, \dots$ be the sequence of configurations of that run. It may be finite or infinite. We define an infinite word $\rho \in \Gamma^\omega$ describing this sequence. If the run is infinite, $\rho = c_0\#c_1\#c_2\#\dots$. Otherwise, if c_k is the last configuration, $\rho = c_0\#c_1\#\dots c_k(\#c_k)^\omega$.

Since all configurations have length n and are determined by the transitions of the machine, there is a function $R : \Gamma^3 \to \Gamma$ such that for all $i \in \mathbb{N}_{>0}$, $\rho[i+n+1] = R(\rho[i-1], \rho[i], \rho[i+1])$. Note that this is the case even when the run is finite.

We apply the morphism ϕ to ρ to obtain a word w on $\{0,1\}$. From now on we use the term *bit* to mean a single letter of w, and *position* to mean the sequence of m bits between positions im and $(i+1)m-1$, for some $i \in \mathbb{N}$. We say that a wins word is well-shaped if every position is either 0^m or $\phi(\gamma)$ for some γ.

We construct a labelled graph that computes w as follows. We say that the graph reads a letter γ if it goes through a sequence of edges with labels $-^{\beta(\gamma)}\text{✗}-^{m-\beta(\gamma)-1}$. Assuming the current wins word is well-shaped, this is only possible if it has $\phi(\gamma)$ at this position. The graph writes $0^{\beta(\gamma)}10^{m-\beta(\gamma)-1}$ means going through a sequence of edges with labels $-^{\beta(\gamma)}\text{✓}-^{m-\beta(\gamma)-1}$. If the position was 0^m or $\phi(\gamma)$ before, it is $\phi(\gamma)$ afterwards. The graph skips a position if it goes through a sequence of m edges labelled $-$. We say that the graph writes *win* (resp. reads *win*) on a position if it goes through m edges labelled ✓ (resp. ✗).

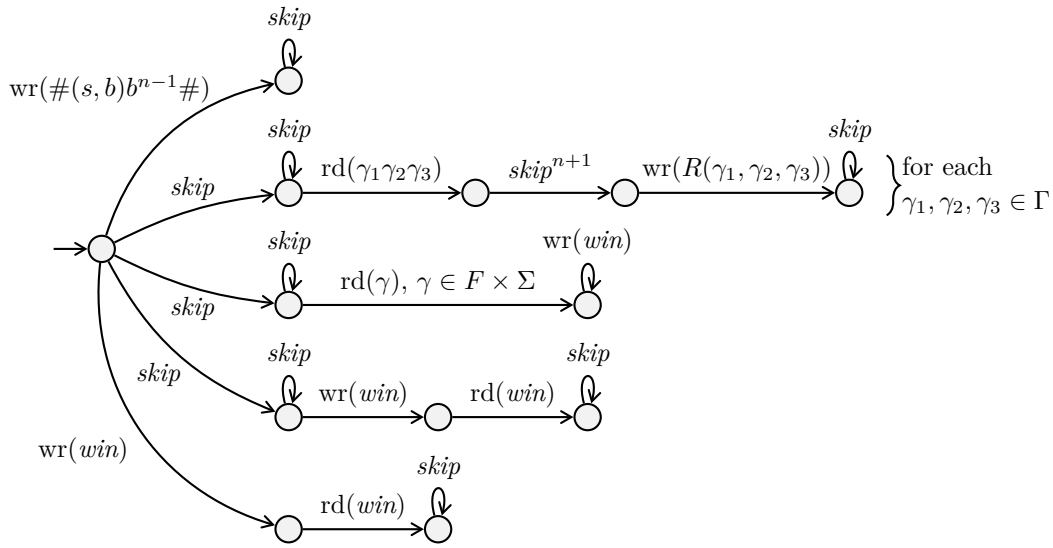


Figure 9 The machine for the PSPACE-hardness reduction. Here wr and rd describe writing and reading actions, while *skip* (resp. $skip^{n+1}$) stands for a sequence of m edges (resp. $(n+1)m$).

The graph can do four things, illustrated in Figure 9:

- It can write $\#(s,b)b^{n-1}\#$ (first branch in the figure)
- For each $\gamma_1, \gamma_2, \gamma_3 \in \Gamma$, it can read $\gamma_1\gamma_2\gamma_3$ at some point in the word, skip $n+1$ positions and write $R(\gamma_1, \gamma_2, \gamma_3)$ (second branch in the figure).
- It can read a letter of $F \times \Sigma$ and write *win* on all following positions (third branch in the figure).
- It can write *win* and read *win* on the next position (two last branches in the figure: the fifth branch lets us apply this on the first position and the fourth branch on the others positions).

While the last two items are not applied, all that we can do is apply the first and second items. It is clear from the construction that the resulting wins word stays well-shaped. Further, the resulting word will always be less or equal to w for the $\preceq$ partial ordering: this results from the determinism of the machine: for every position there is only one letter that we can write on it.

If w contains a letter of $F \times \Sigma$ at some position, we will eventually write it, and apply the third item to set all following bits to 1, and the last item repeatedly for the rest of the bits.

Otherwise, we will only obtain wins words with $\phi(\#)$ on their first position, and thus some of the m first bits will stay 0 forever. Hence there is a winning strategy if and only if the run of the Turing machine reaches a final state. ◀