Games with Winning Conditions of High Borel Complexity

Olivier Serre

Games With Winning Conditions of High Borel Complexity

Olivier Serre

LIAFA, Université Paris VII, CNRS
2, place Jussieu, case 7014
F-75251 Paris Cedex 05

Abstract

We first consider infinite two-player games on pushdown graphs. In previous work, Cachat, Duparc and Thomas [4] have presented a winning decidable condition that is Σ_3-complete in the Borel hierarchy. This was the first example of a decidable winning condition of such Borel complexity. We extend this result by giving a family of decidable winning conditions of arbitrary finite Borel complexity. From this family, we deduce a family of decidable winning conditions of arbitrary finite Borel complexity for games played on finite graphs. The problem of deciding the winner for these conditions is shown to be non-elementary.

Key words: Pushdown Automata, Two-player Games, Borel Complexity.

1 Introduction

Infinite two-player games have been intensively studied in the last few years. One of the main motivations is the strong relation that exists with verification questions and controller synthesis. For instance, μ-calculus model checking for finite graphs (respectively for pushdown graphs) is polynomially equivalent to the problem of deciding the winner in a game played on a finite graph [7] (respectively on a pushdown graph [24]). In addition, constructing a winning strategy is the same as synthesizing a discrete controller [1].

⋆ This research has been partially supported by the European Community Research Training Network "Games and Automata for Synthesis and Validation" (GAMES), (contract HPRN-CT-2002-00283), see www.games.rwth-aachen.de.

One important branch of game theory is developed in the framework of descriptive set theory in which the central question is determinacy, that is, the existence of a winning strategy. One of the deepest results is due to Martin [15] and states that, for Borel winning conditions, games are determined. In computer science, the games considered are in general equipped with winning conditions of low Borel complexity and therefore trivially determined. Nevertheless, deciding the winner is, in many cases, a difficult problem. Since we are mostly interested in decidable games, it is natural to ask whether there exist decidable games of arbitrary high finite Borel complexity.

For finite graphs and for the natural conditions appearing in verification and model-checking, efficient algorithms are known to decide the winner and to compute the associated winning strategies [20,25,10,22]. These winning conditions all belong to a low level of the Borel hierarchy, namely to the boolean closure of the Borel class Σ_2.

In [21], Thomas proposes to study games with winning conditions of Borel level larger than 2. In the paper, we focus on this topic and exhibit a family of winning conditions on pushdown games that have an arbitrary high Borel complexity while remaining decidable. As a corollary, one obtains a similar result for games on finite graphs. In addition, these games have effective winning strategies.

The first results concerning high level Borel conditions come from pushdown games. In this model, the game graph is the infinite graph of the configurations of a pushdown process. Walukiewicz has shown that parity games on such graphs can be effectively solved [24]. For this model, winning conditions exploiting the infinity of the stack become natural. In [4], Cachat, Duparc and Thomas have considered the following condition: Eve wins a play if and only if some configuration is infinitely often visited. They have shown that it is a decidable winning condition belonging to Σ_3. More recently, Bouquet, Serre and Walukiewicz have considered in [2] winning conditions that are boolean combinations of a Büchi condition with a condition called *unboundedness* that requires the stack to be unbounded. Gimbert has used the elegant method from [12] to study the boolean combination of a parity condition with unboundedness [9]. All these winning conditions are closely related to the one of [4] and remain decidable [2,9]. A natural question was therefore to consider higher level winning conditions.

In this paper, we give a uniform answer to the question of [21] by providing a family of winning conditions of increasing finite Borel complexity. The main idea is to require the stack to converge to some limit and then to have additional conditions on the limit. To solve classical conditions on pushdown games, one method consists in reducing the problem to a game on a finite graph [24,2]. We adapt this method and reduce the problem of deciding the

winner in a pushdown game to the problem of deciding the winner in another pushdown game, equipped with a lower winning condition. Therefore, the proof goes by induction.

From the proofs we also infer the effectiveness of the winning strategies. Whereas for previously studied winning conditions on pushdown games, the set of winning positions was regular, it is no longer the case here. The exact nature of these sets remains open. We further show that the complexity of determining the winner for these high level Borel winning conditions is non-elementary and is elementary-hard. We also show that Eve has, from a winning position, a persistent strategy, that is, a strategy using memory but such that the move given from some vertex is always the same for a given play.

The paper is organized as follows. In Section 2, we start with basic definitions on games and introduce the family of winning conditions that we will consider in the rest of the paper. In Section 3, we precisely characterize the Borel complexity of these winning conditions. In Section 4, we give the decidability results and constructions of these games. Remark that it gives a new proof of the decidability results of [4]. In Section 5, we show that the complexity of deciding the winner for such winning conditions is non-elementary and is elementary-hard. Finally, in Section 6, we discuss several points.

2 Definitions

2.1 Basic Definitions

An *alphabet* A is a finite or infinite set of letters. A^* denotes the set of *finite words* on A, A^ω the set of *infinite words* on A and A^∞ the set $A^* \cup A^\omega$. The empty word is denoted by ε. For a word u, we denote its (possibly infinite) length by $|u|$. For $i < |u|$, we write $u(i)$ for the i-th letter of u.

Let $u \in A^*$ and $v \in A^\infty$. Then u is a *prefix* of v, denoted $u \sqsubseteq v$ if there exists some word $w \in A^\infty$ such that $v = u \cdot w$. For any word $u \in A^\infty$, there exists a unique prefix of length k for all $k \leq |u|$. This prefix is denoted by $u\restriction_k$.

Definition 1 (Limit of a sequence of finite words) *Let $(u_i)_{i\geq 0} \in (A^*)^\mathbb{N}$ be an infinite sequence of words. The limit $\lim_{i\in\mathbb{N}} u_i$ of $(u_i)_{i\geq 0}$ is the maximal word satisfying the following: for each j, there exists an index r such that the j-th letter of $\lim_{i\in\mathbb{N}} u_i$ equals the j-th letter of u_p for every $p \geq r$. Note that $\lim_{i\in\mathbb{N}} u_i$ can be either finite or infinite.*

We recall now the classical notion of deterministic pushdown automaton.

A *deterministic pushdown automaton* with input from A^∞ is a tuple $\mathcal{A} = \langle Q, \Gamma, A, \bot, q_{in}, \delta \rangle$, where Q is a finite set of states, Γ is a finite set of stack symbols, $\bot \in \Gamma$ is a special bottom-of-stack symbol, A is the finite input alphabet, $q_{in} \in Q$ is the initial state and

$$\delta : Q \times \Gamma \times A \to \{skip(q), pop(q), push(q, \gamma) \mid q \in Q,\ \gamma \in \Gamma \setminus \{\bot\}\}$$

is the transition function. In addition, we require that for all $q, q' \in Q$, $a \in A$, $\delta(q, \bot, a) \neq pop(q')$ (the bottom-of-stack symbol is never removed).

A *stack* is an element of the set $St = (\Gamma \setminus \{\bot\})^* \cdot \bot$. A configuration of $\mathcal{A}$ is a pair (q, σ) with $q \in Q$ and $\sigma \in St$. Note that the top stack symbol in some configuration (q, σ) is the leftmost symbol of σ.

A configuration $(q, \gamma\sigma)$, for some $q \in Q$ and $\gamma\sigma \in St$, has a unique *successor* by some letter $a \in A$, which is defined as follows, depending on $\delta(q, \sigma, a)$:

- If $\delta(q, \gamma, a) = skip(q')$, it is $(q', \gamma\sigma)$.
- If $\delta(q, \gamma, a) = pop(q')$, it is (q', σ).
- If $\delta(q, \gamma, a) = push(q', \gamma')$, it is $(q', \gamma'\gamma\sigma)$.

A *run* of $\mathcal{A}$ on a (possibly infinite) word $\alpha = \alpha_0\alpha_1 \cdots$ starts from the configuration $(q_{in}, \bot)$. $\mathcal{A}$ reads α_0 and goes to the successor of $(q_{in}, \bot)$ by α_0, then it reads α_1 and goes to the successor of the current configuration by α_1, and so on.

One can in addition equip such an automaton with a classical acceptance condition, for instance a parity condition. In that case, one has a mapping *col* from Q to a finite set of colors $C \subset \mathbb{N}$. This coloring function naturally extends to the set of configurations by setting $col((q, \sigma)) = col(q)$. Finally, an infinite word α is accepted by $\mathcal{A}$, if and only if the smallest color appearing infinitely often in the run of $\mathcal{A}$ on α is even.

If $\mathcal{A}$ is a deterministic pushdown automaton equipped with a parity acceptance condition, we denote $L(\mathcal{A})$ the language *accepted* by $\mathcal{A}$.

Let $\mathcal{A}$ be a deterministic pushdown automaton and let α be some infinite word on the input alphabet of $\mathcal{A}$. We say that the stack of $\mathcal{A}$ is *strictly unbounded* when reading α, if the sequence $(\sigma_i)_{i \geq 0}$ of stack contents in the run of $\mathcal{A}$ on α is such that $\lim_{i \in \mathbb{N}} \sigma_i$ is infinite. Equivalently, we require that for all $h \geq 0$, there is some index j_h such that $|\sigma_i| \geq h$ for all $i \geq j_h$, and the limit of the stack is the infinite word $\sigma_{j_0}(0)\sigma_{j_1}(1)\sigma_{j_2}(2) \cdots$.

Now, let $n \geq 0$ be some integer and let us consider a collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of deterministic pushdown automata (if $n = 0$ this collection is considered to be empty). Let $\mathcal{A}_{n+1}$ be a deterministic pushdown automaton equipped with a parity acceptance condition. On input alphabet of $\mathcal{A}_1, \ldots, \mathcal{A}_{n+1}$, we require the following *stack consistency* property:

For all $1 \leq i \leq n$, the input alphabet of $\mathcal{A}_{i+1}$ is the stack alphabet of $\mathcal{A}_i$.

Let A be the input alphabet of $\mathcal{A}_1$. We associate with $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ a language of infinite words on the alphabet A, that we denote $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$, and which is defined as follows:

- If $n = 0$, $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}) = L(\mathcal{A}_{n+1})$ is the language accepted by $\mathcal{A}_{n+1}$.
- If $n > 0$, $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ is the set of infinite words α_0 on the alphabet A such that:
  - When $\mathcal{A}_1$ reads α_0, its stack is strictly unbounded and therefore the sequence of stack contents converges to some limit α_1.
  - $\alpha_1 \in L(\mathcal{A}_2 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$.

Equivalently, a word $\alpha_0 \in A^\omega$ belongs to $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ if and only if:

- For all $1 \leq i \leq n$, when $\mathcal{A}_i$ reads α_{i-1}, its stack is strictly unbounded and the sequence of stack contents converges to some limit α_i.
- $\mathcal{A}_{n+1}$ accepts α_n.

Figure 1 illustrate the case where $n = 3$.

Finally, we denote by $\mathbb{C}_n(A)$ the class of languages L on some finite alphabet A such that $L = (\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ for some collection of deterministic pushdown automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and some deterministic parity automaton $\mathcal{A}_{n+1}$. In particular $\mathbb{C}_0(A)$ is the class of deterministic ω-context free languages on the alphabet A [5].

In the sequel, when considering pushdown automata $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ used to define a language as described above, we will implicitly suppose that they are stack consistent.

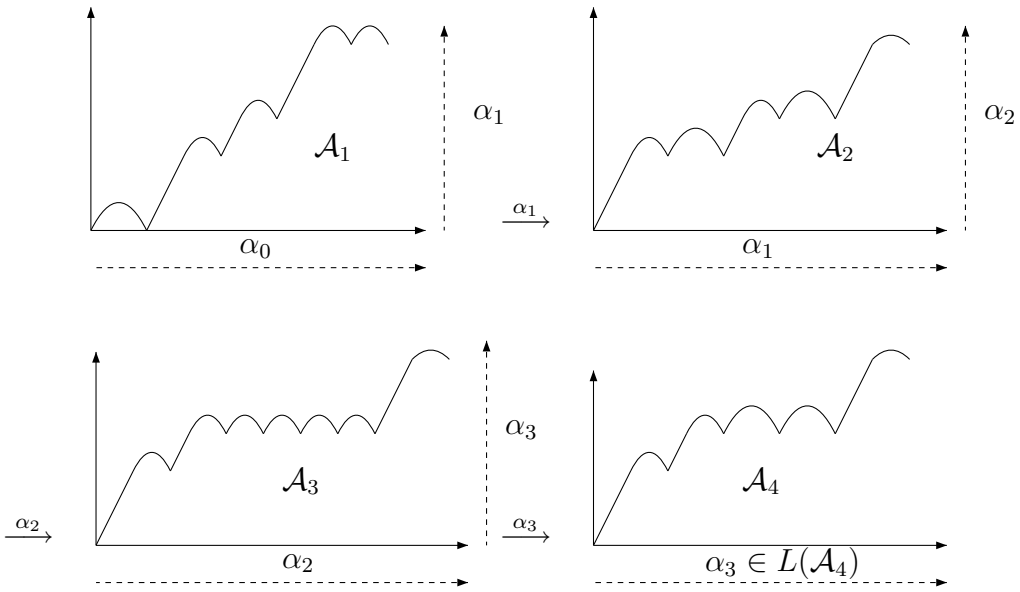


Fig. 1. The language $L(\mathcal{A}_1 \rhd \mathcal{A}_2 \rhd \mathcal{A}_3 \rhd \mathcal{A}_4)$

2.3 Games

Let A be a finite alphabet and let $G = (V, E)$ be a graph with edges labeled by letters in some alphabet A or by the empty word ε, that is $E \subseteq V \times (A \cup \{\varepsilon\}) \times V$. Let $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ be a partition of V between two players Eve and Adam. A *game graph* is such a tuple $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$.

A two-player game on a game graph $\mathcal{G}$ is a pair $\mathbb{G} = (\mathcal{G}, \Omega)$, where Ω is a *winning condition*, which can be of two kinds:

- Ω is an *internal winning condition* if $\Omega \subseteq V^\omega$.
- Ω is an *external winning condition* if $\Omega \subseteq A^\infty$.

A *play* from some vertex v_0 proceeds as follows: if $v_0 \in V_{\mathbf{E}}$, Eve chooses a successor v_1 and an edge $(v_0, a_0, v_1) \in E$. Otherwise, it is Adam's turn to choose a successor v_1 and an edge. If there is no such v_1, then the play ends in v_0, otherwise the player to whom v_1 belongs tries to move to some v_2 and so on. Therefore a play starting from v_0 is a finite or infinite sequence $\lambda = v_0 a_0 v_1 a_1 v_2 \cdots \in V((A \cup \{\varepsilon\})V)^\infty$ such that $(v_i, a_i, v_{i+1}) \in E$ for all i. In the case where the play is finite, we require that there is no $(v_n, a_n, v_{n+1}) \in E$, if v_n was the last vertex of the play. A *partial play* is any prefix of a play.

A finite play is lost by the player that cannot move. An infinite play $\lambda = v_0 a_0 v_1 a_1 v_2 \cdots$ is won by Eve if and only if:

- Ω is internal and $v_0 v_1 v_2 \cdots \in \Omega$.
- Ω is external and $a_0 a_1 a_2 \cdots \in \Omega$.

For a play $\lambda = v_0 a_0 v_1 a_1 v_2 \cdots$ we denote by $\mathrm{Lab}(\lambda)$ the word $a_0 a_1 \cdots$. For instance $\mathrm{Lab}(v_0 \varepsilon v_1 b v_2 b v_3 \varepsilon v_4 \varepsilon v_5 a v_6 \varepsilon v_7) = bba$. Therefore, if λ is a play in a game equipped with an external winning condition Ω, it is won by Eve if and only if $\mathrm{Lab}(\lambda) \in \Omega$.

A strategy for Eve is a function assigning, to any partial play ending in some vertex $v \in V_{\mathbf{E}}$, an edge $(v, a, v') \in E$. Eve *respects a strategy* f during some play $\lambda = v_0 a_0 v_1 a_1 v_2 \cdots$ if $(v_i, a_i, v_{i+1}) = f(v_0 a_0 \cdots v_i)$, for all $i \geq 0$ such that $v_i \in V_{\mathbf{E}}$. Finally a strategy for Eve is *winning* from some position v, if any play starting from $v \in V$, where Eve respects f, is won by her. A vertex $v \in V$ is winning for Eve if she has a winning strategy from v. Symmetrically, one defines the strategies and the winning positions for Adam.

A game $\mathbb{G}$ is *determined* if, from any position, either Eve or Adam has a winning strategy.

2.4 Pushdown Games

Pushdown processes provide a natural model for programs with recursive procedures. They are like pushdown automata except that they are nondeterministic. In addition, in this model, the input word (and therefore the initial state and acceptance condition) are ignored.

More formally, a *pushdown process* is a tuple $\mathcal{P} = \langle Q, \Gamma, \bot, \Delta \rangle$ where Q is a finite set of states, Γ is a finite stack alphabet that contains a special bottom-of-stack symbol $\bot$ and

$$\Delta : Q \times \Gamma \to 2^{(\{skip(q), pop(q), push(q,\gamma) | q \in Q, \gamma \in \Gamma \setminus \{\bot\}\})}$$

is the transition relation. As for pushdown automata, we require that, for all $q \in Q$, $\Delta(q, \bot)$ does not contain any element $pop(q')$. Finally, the notions of stack and configuration of a pushdown process are defined as for pushdown automata.

From $\mathcal{P}$, one defines an infinite graph, denoted $G = (V, E)$, whose vertices are the configurations of $\mathcal{P}$, and edges E are defined by the transition relation Δ, i.e., from a vertex $(p, \gamma\sigma)$ one has:

- $(q, \gamma\sigma)$ whenever $skip(q) \in \Delta(p, \gamma)$.
- (q, σ) whenever $pop(q) \in \Delta(p, \gamma)$.
- $(q, \gamma'\gamma\sigma)$ whenever $push(q, \gamma') \in \Delta(p, \gamma)$.

Finally, let $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ be some partition of Q between Eve and Adam. It induces a natural partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ of V by setting $V_{\mathbf{E}} = Q_{\mathbf{E}} \times St$ and $V_{\mathbf{A}} = Q_{\mathbf{A}} \times St$. The game graph $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ is called a *pushdown game graph*.

Note that in a pushdown game graph, the edges are not labeled. Therefore, we will equip them only with internal winning conditions. Moreover a play will be represented as a word on the alphabet V of vertices, and a strategy for Eve will be a function $f : V^*V_{\mathbf{E}} \to V$.

For a vertex $v = (q, \sigma)$ of V, we define $|v|$ to be the length of σ. In a play $\lambda = v_0 v_1 v_2 \cdots$, the stack is *strictly unbounded* if the stack size converges to $+\infty$. More formally we require that for all $k \geq 0$, there exists some i such that: for all $j \geq i$, $|v_j| > k$.

If the stack in a play $\lambda = v_0 v_1 \ldots$ is strictly unbounded, we will consider its *limit*, $\mathrm{StLim}\lambda = \lim \sigma_i$, where for all $i \geq 0$, $v_i = (p_i, \sigma_i)$ for some $p_i \in Q$ and $\sigma_i \in St$.

The following internal winning condition $\Omega_{ubd} = \{\lambda \mid \text{the stack is strictly unbounded in } \lambda\}$ is called the *strict unboundedness winning condition*. A pushdown game $\mathbb{G} = (\mathcal{G}, \Omega_{ubd})$ is called a *strict unboundedness pushdown game*.

Remark 2 *In [4], it is shown that it can be decided in* DEXPTIME *whether Eve has a winning strategy in a pushdown game equipped with a winning condition requiring that some configuration is infinitely often visited. It is easily seen that this condition is the dual condition of the strict unboundedness winning condition. Therefore, it is equivalent to decide a strict unboundedness pushdown game. In section 4.3, we show that our main result gives the one of [4] as a corollary, hence provides a new proof.*

Finally, let us mention the *parity condition* on pushdown games. Let *col* be a coloring function from Q into a finite set of colors $C \subset \mathbb{N}$. This function is easily extended into a function from V in C by setting $col((q, \sigma)) = col(q)$. The parity condition is the internal winning condition defined by:

$$\Omega_{par} = \{v_0 v_1 \cdots \mid \inf\{c \mid \exists^\infty i \text{ s.t. } col(v_i) = c\} \text{ is even}\}$$

2.5 The Winning Conditions $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ and $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$

Let $n \geq 0$ be some integer and let $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ be some pushdown automata where in addition $\mathcal{A}_{n+1}$ is equipped with a parity condition. Let Γ be the input alphabet of $\mathcal{A}_1$. From $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$, we define two winning conditions, an external one that will be used for games played on finite graphs, and an internal one for pushdown games. It will be shown later that they are

closely related.

Definition 3 *The external winning condition* $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$*, is defined by:*

$$\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}} = L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$$

Definition 4 *Let* $\mathcal{G}$ *be a pushdown game graph constructed from some pushdown process with* Γ *as stack alphabet. Then* $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ *is defined by:*

$$\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}} = \{\lambda \in \Omega_{ubd} \mid \lim \lambda \in L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})\}$$

Example 5 *We finish with examples of such winning conditions:*

(1) Let $\mathcal{A}$ *be some parity automaton accepting all infinite words on some alphabet* Γ*. Then the winning condition* $\Omega^{int}_{\mathcal{A}}$ *is the strict unboundedness winning condition.*

(2) Consider the deterministic parity pushdown automaton $\mathcal{A}$ *that accepts the language* $\{\alpha \in \{p, c, t\}^\omega \mid \forall k \geq 0,\ |\alpha\restriction_k|_c \leq |\alpha\restriction_k|_p\}$ *of words* α *on the alphabet* $\{c, p, t\}$ *such that in any prefix of* α*, the number of* p *is greater than the number of* c *(*$|u|_a$ *designates the number of occurrences of some letter* a *in some word* u*). If* p *stands for* produced*,* c *for* consumed *and* t *for* transform*, the winning condition* $\Omega^{int}_{\mathcal{A}}$ *expresses that, in a system using recursive procedure and such that at the end of the main procedure (where all recursive calls end), some resource may be produced, consumed or transformed (which is also recorded in the stack by pushing either* p*,* c *or* t*), there is always an available resource when the consumer asks for it.*

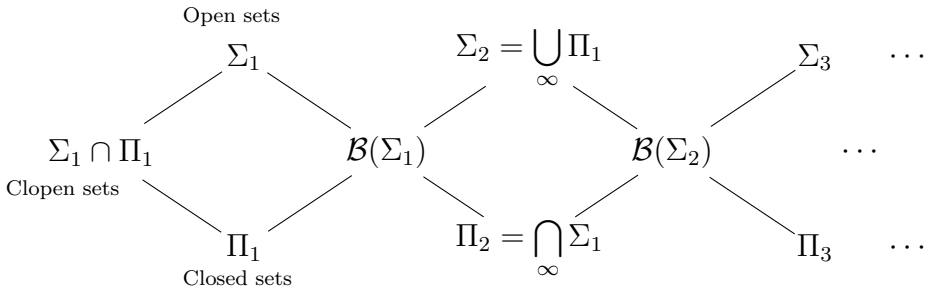
3 Borel Complexity

3.1 Borel Hierarchy

Let A be a (possibly infinite) alphabet. We consider the set A^ω of infinite words on the alphabet A, and we equip it with the usual Cantor topology where the open sets are those of the form $W \cdot A^\omega$ where $W \subseteq A^*$ is a language of finite words on the alphabet A. The *finite Borel hierarchy* $(\Sigma_1, \Pi_1), (\Sigma_2, \Pi_2), \cdots$ is inductively defined as follows:

- $\Sigma_1 = \{W \cdot A^\omega \mid W \subseteq A^*\}$ is the set of open sets.
- For all $n \geq 1$, $\Pi_n = \{S^{co} \mid S \in \Sigma_n\}$ consists of the complements of Σ_n-sets.
- For all $n \geq 1$, $\Sigma_{n+1} = \left\{ \bigcup_{i \in \mathbb{N}} S_i \mid \forall i \in \mathbb{N},\ S_i \in \Pi_n \right\}$ is the set of countable union of Π_n-sets.

Finally, if we denote by $\mathcal{B}(\Sigma_n)$ the Boolean combination of Σ_n-sets, we have the following strict inclusions:



A set S is a *proper-Σ_n-set* if it is in Σ_n but not in Π_n.

3.2 Borel Complexity of a Winning Condition

Let $\mathbb{G} = (\mathcal{G}, \Omega)$ be some game played on a game graph $\mathcal{G}$ and equipped with a winning condition Ω. The Borel complexity of the winning condition Ω is its Borel complexity when considered as a set on the alphabet of the vertices of $\mathcal{G}$ if Ω is internal, and on the alphabet A that labels the edges of $\mathcal{G}$ if Ω is external.

Here are some examples of internal winning conditions:

Example 6 *(1) Consider a reachability winning condition (Eve wins if and only if the play eventually visits a vertex in some subset $F \subseteq V$). Such a condition is a Σ_1-winning condition, as the winning condition for Eve is $(V^*F)V^\omega$.*

(2) Consider a Büchi winning condition (Eve wins if and only if the play infinitely visits vertices belonging to some subset $F \subseteq V$). Such a condition is a Π_2-winning condition, as the winning condition for Eve is $\bigcap_{n\geq 0}[(V^nV^*F)V^\omega]$.

(3) Consider a Muller condition (Eve wins if and only if the set of infinitely visited vertices belongs to a set of subsets of V). Such a winning condition is a $\mathcal{B}(\Sigma_2)$-winning condition, as it is a Boolean combination of Büchi winning conditions.

(4) Consider an unboundedness winning condition for pushdown games (Eve wins if and only if the stack size is not bounded). Such a condition is a Π_2-winning condition. Indeed, for any $n \geq 0$, if one denotes by V_n the set of configurations of stack size n, the winning condition for Eve is $\bigcap_{n\geq 0}[(V^*V_n)V^\omega]$.

(5) Consider a strict unboundedness winning condition for pushdown games (Eve wins if and only if the stack size converges to ∞). The corresponding condition for Adam is the one considered in [4]: Adam wins if and only

if some configuration (equivalently some stack size) is infinitely repeated. Therefore, if one denotes by V_n the set of configurations of stack size n, the winning condition for Adam is

$$\bigcup_{n\geq 0}\bigcap_{m\geq 0}[(V^mV^*V_n)V^\omega]$$

Therefore, the winning condition for Adam is a Σ_3-set, and thus the strict unboundedness winning condition is a Π_3-winning condition for Eve.

Finally, let us mention the well-known Martin's Borel determinacy theorem.

Theorem 7 *[15] Any game equipped with a Borel winning condition is determined.*

3.3 Wadge Games

Definition 8 (Wadge game) *[23] Let A and B be two (possibly infinite) alphabets. Let $X \subseteq A^\omega$ and $Y \subseteq B^\omega$. The Wadge game $G(X,Y)$ is a two-player game between Alice and Bob. Alice first chooses a letter a_0 in A. Then, Bob chooses a (possibly empty) finite word $b_0 \in B^*$. Then Alice chooses a letter a_1, and Bob a word b_1, and so on. Therefore a play consists in writing an infinite word $\alpha = a_0a_1\cdots$ for Alice, and writing a word $\beta = b_0b_1\cdots$ for Bob. Bob wins if and only if both β is infinite, and $\alpha \in X \Leftrightarrow \beta \in Y$.*

Notions of strategies, and winners in Wadge games are defined similarly. These games are strongly related to the following notion.

Definition 9 (Wadge reduction) *We say that $X \subseteq A^\omega$ Wadge reduces to $Y \subseteq B^\omega$, denoted $X \leq_W Y$, if and only if there exists a continuous function $f : A^\omega \to B^\omega$ such that $X = f^{-1}(Y)$. If $X \leq_W Y$ and $Y \leq_W X$, then we say that X and Y are Wadge equivalent and we denote this by $X \equiv_W Y$.*

Then, we have the following well-known result.

Proposition 10 ([23]) *Bob has a winning strategy in the game $G(X,Y)$ if and only if $X \leq_W Y$.*

Example 11 *Consider the language A^ω on some non empty alphabet A. Then $A^\omega \leq_w Y$ for any non empty set Y. Indeed, a winning strategy for Bob in $W(A^\omega, Y)$ consists in describing some word in Y.*

Let $A = \{a,b\}$. Let $X \subseteq A^\omega$ be a closed set and let $Y = \{a^\omega\} \subseteq A^\omega$. Then, $X \leq_w Y$. Indeed, Bob plays a if the prefix played by Alice is a prefix of some word in X. Otherwise, he plays b. This strategy is winning as an infinite word

is in a closed set X if and only if any prefix of the word is a prefix of some word in X (see [17]). Note that these two examples stress the importance of the underlying alphabets.

The Wadge equivalence preserves the Borel hierarchy levels.

Proposition 12 *Let X and Y be two Wadge equivalent sets. Then they belong to the same level of the Borel hierarchy.*

Then, it is natural to consider the following completeness notion induced by the relation $\leq_W$.

Definition 13 (Complete sets) *A set $Y \in \Sigma_n$ is Σ_n-complete if and only if $X \leq_W Y$ for all $X \in \Sigma_n$. In particular, a Σ_n-complete set is a proper Σ_n-set. One easily defines Π_n-complete sets.*

Remark 14 *The notion of complete sets is not relevant for the class $\mathcal{B}(\Sigma_n)$, as there are no complete set for such a class for $n \geq 1$ [11].*

Now, we give some examples to illustrate the completeness notion.

Example 15 *Let $A = \{a, b\}$. Let $X \subseteq A^\omega$ be the set of infinite words that contains infinitely many a. X is a Π_2-complete set. Indeed $X = \bigcap_{i \geq 0} A^i A^* a A^\omega$, hence X is a Π_2-set. Let Y be a Π_2-set on some alphabet B. Therefore, $Y = \bigcap_{i \geq 0} Y_i$ for some family $(Y_i)_{i \geq 0}$ of open sets, where $Y_i = Z_i B^\omega$. In the game $G(Y, X)$, Bob has a winning strategy that consists in maintaining some counter i which is initialized to 0. If the word already written by Alice is in $Z_i B^*$, he plays a and changes his counter to $i + 1$. Otherwise, he plays b and does not change the value of his counter. Therefore, the word played by Bob contains infinitely many a if and only if the word played by Alice belongs to Y_i for all $i \geq 0$, that is, it belongs to Y.*

3.4 The Operation $X \mapsto X^\sim$

In [6], Duparc introduces several Borel operations that are homomorphic to ordinal sum, to multiplication by a countable ordinal and to ordinal exponentiation of base κ (for some uncountable regular cardinal κ). Here, we only focus on the operation $X \mapsto X^\sim$, which is the Borel counterpart of the ordinal exponentiation of base κ.

The operation $X \mapsto X^\sim$ works on sets $X \subseteq A^\infty$ of finite or infinite words on some alphabet A. Nevertheless one needs to transform finite words into infinite words to define the operation. In this paper, we will only use a consequence

of Duparc's results that works for languages of infinite words. That is why we only describe the result in the framework of languages of infinite words.

Definition 16 (def. 22 of [6]) *Let A be some alphabet, let $X \subseteq A^\infty$, and $\twoheadleftarrow \notin A$ (a symbol for "Back Space"), then $X^\sim = \{u \in (A \cup \{\twoheadleftarrow\})^\infty \mid u^{\leftarrowtriangle} \in X\}$ where $u^{\leftarrowtriangle}$ is inductively defined by:*

- $\varepsilon^{\leftarrowtriangle} = \varepsilon$
- *for u finite with $|u^{\leftarrowtriangle}| = k$:*
  - $(u \cdot a)^{\leftarrowtriangle} = u^{\leftarrowtriangle} \cdot a$ *if* $a \neq \twoheadleftarrow$.
  - $(u \cdot \twoheadleftarrow)^{\leftarrowtriangle} = u^{\leftarrowtriangle}{\restriction}_{(k-1)}$ *if* $k > 0$ *(erases the last letter of $u^{\leftarrowtriangle}$).*
  - $(u \cdot \twoheadleftarrow)^{\leftarrowtriangle} = \varepsilon$ *if* $k = 0$ *(there is nothing to erase).*
- *for u infinite, $u^{\leftarrowtriangle} = \lim_{n\in\mathbb{N}}((u{\restriction}_n)^{\leftarrowtriangle})$.*

For instance, $ab \twoheadleftarrow\twoheadleftarrow c = a \twoheadleftarrow c = c$, $bb(ab \twoheadleftarrow)^\omega = bba^\omega$ and $bb(b \twoheadleftarrow)^\omega = bb$.

The operation $X \mapsto X^\sim$ has a very natural interpretation in terms of Wadge game. A player in charge of $X^\sim$ is like a player in charge of X that can in addition erase symbols as often as he wants by simply playing the Back Space letter $\twoheadleftarrow$.

The iterated version of the operation $X \mapsto X^\sim$ is defined as follows.

Definition 17 *Let X be some set. Then, we define $X^{\sim 0} = X$ and $\forall n \in \mathbb{N}$, $X^{\sim n+1} = (X^{\sim n})^\sim$.*

In this paper, we use the following consequence of Lemma 31 of [6], that works for languages of infinite words.

Lemma 18 *[8] Let A be some alphabet and let $X \subseteq A^\omega$ be some Π_k-complete set for some $k \geq 2$. Then, $(X^{\sim n})$ is a Π_{n+k}-complete set, for all $n \geq 0$.*

Finally, let us give the following result due to Löding [13].

Lemma 19 *Let A be some alphabet and let $X \subseteq A^\omega$. If $X \in \mathcal{B}(\Sigma_n)$ for $n \geq 2$ then $L^\sim \in \mathcal{B}(\Sigma_{n+1})$.*

PROOF. We start proving the result for open sets. For this, it suffices to prove it for some Σ_1-complete set as the operation $X \mapsto X^\sim$ respects the Wadge ordering $\leq_W$ [6]. Let us consider the language $O = (a^*b)A^\omega$ on the alphabet $A = \{a, b\}$ which is the complement of the Π_1-complete language a^ω (see Example 11).

Then one has:

$$O^{\sim} = \bigcup_{n \geq 0} ((A \cup \{\twoheadleftarrow\})^n b (A \cup \{\twoheadleftarrow\})^{\omega} \cap K_n$$

where K_n is the set of infinite words α on the alphabet $A \cup \{\twoheadleftarrow\}$ such that the n-th letter of α is not erased when computing $\alpha^{\leftarrow\rho}$. More precisely,

$$K_n = (H_{\geq n}(A \cup \{\twoheadleftarrow\})^{\omega})^{co}$$

where $H_{\geq n}$ is the set of finite words u such that the n-th letter of u is erased when computing $n^{\leftarrow\rho}$, that is $u(n+1) \cdots u(|n|-1) = v \twoheadleftarrow v'$ for some v such that $v^{\leftarrow\rho} = \varepsilon$.

Therefore, K_n is a Π_1-set and therefore $O^{\sim}$ is a Σ_2-set.

Now, note that the operation $X \mapsto X^{\sim}$ on set of infinite words distributes over intersection and union.

For complementation, in general, $(X^{co})^{\sim} \subsetneq (X^{\sim})^{co}$. Indeed, $(X^{\sim})^{co}$ also contains the words α such that $\alpha^{\leftarrow\rho}$ is finite. Therefore $(X^{co})^{\sim} = (X^{\sim})^{co} \cap Inf$, where Inf is the set of words $\alpha \in (A \cup \{\twoheadleftarrow\})^{\omega}$ such that $\alpha^{\leftarrow\rho}$ is infinite:

$$Inf = \bigcap_{m \geq 0} \bigcup_{n \geq m} K_n$$

Hence, Inf is a Π_3-set.

Now, if X is some set in $\mathcal{B}(\Sigma_n)$ for some $n \geq 2$, one applies the $X \mapsto X^{\sim}$ operation to the formula proving its membership to $\mathcal{B}(\Sigma_n)$, pushes the preceding operation to the level of Σ_1-sets and intersect with Inf whenever complementation is used. In the resulting formula, all Σ_1-sets have been changed into Σ_2-sets, and therefore it shows that $X^{\sim}$ belongs to $\mathcal{B}(\Sigma_{n+1})$ (intersection with Inf does not increase the Borel complexity, as $n \geq 2$). $\square$

3.5 Borel Complexity of $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$

In this section, we study the Borel complexity of languages in the class $\mathbb{C}_n(A)$ for some finite alphabet A and some integer $n \geq 0$.

More precisely, we show the following result.

Theorem 20 *Let A be some finite alphabet and let $n \geq 0$ be some integer. Then a language in $\mathbb{C}_n(A)$ belongs to $\mathcal{B}(\Sigma_{n+2})$. In addition, there is a language in $\mathbb{C}_n(A)$ that is Π_{n+2}-complete.*

Note that for $n = 0$ the result is the classical one stating that a language recognized by some deterministic machine equipped with a parity condition is in $\mathcal{B}(\Sigma_2)$ (the proof is a generalization of the one showing that ω-regular languages are in $\mathcal{B}(\Sigma_2)$ [17]). For completeness, one can consider the ω-regular language on the alphabet $\{a, b\}$ of infinite words containing infinitely many occurrences of the letter a (see Example 15).

The proof of Theorem 20 goes by induction on n, and relies on the following lemma.

Lemma 21 *Let $n \geq 1$ and let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be a collection of pushdown automata and let $\mathcal{A}_{n+1}$ be a parity pushdown automaton. Then $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}) \leq_W L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})^\sim$.*

In addition, there is some pushdown automaton $\mathcal{A}_1$ such that the preceding inequality is an equivalence.

PROOF. Let $X = L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ and let $Y = L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})^\sim$.

We show that Bob has a winning strategy in the Wadge game $G(X, Y)$. Indeed, Bob plays so that when Alice has played some word u, he has played a word v such that $v^{\leftarrow\rho}$ is the stack content of $\mathcal{A}_1$ when having read u. Bob wins by the very definition of X from Y, hence $X \leq_W Y$.

Now consider the special case where $\mathcal{A}_1 = \langle \{q\}, \Gamma, \Sigma, \bot, q, \delta \rangle$ where:

- $\Gamma = \Sigma \cup \{\twoheadleftarrow\}$ where $\twoheadleftarrow \notin \Sigma$.
- For all $\gamma \in \Sigma$, $\delta(q, \gamma, \twoheadleftarrow) = pop(q)$ and $\delta(q, \gamma, a) = push(q, a)$, for $a \neq \twoheadleftarrow$.

Then, if we consider $\twoheadleftarrow$ as an eraser, we directly have that the stack contents of $\mathcal{A}_1$, after reading some word u, is $u^{\leftarrow\rho}$. In addition, the stack limit of $\mathcal{A}_1$ after reading some infinite word α is $\alpha^{\leftarrow\rho}$. By definition of acceptance, an infinite word $\alpha \in X$ if and only if the stack limit of $\mathcal{A}_1$ when reading α is infinite and is in $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Therefore, $\alpha \in X$ if and only if $\alpha^{\leftarrow\rho} \in L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. This exactly means that $X \equiv_W Y$. $\square$

Then, Theorem 20 follows from transitivity of $\leq_W$, lemmas 18 and 19, and from the basic case $n = 0$.

In this section, we discuss the Borel complexity of the winning conditions of the form $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ and $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$.

For the external winning condition $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$, we have the following corollary of Theorem 20.

Corollary 22 *For all $n \geq 0$ the following holds:*

- *For any collection of deterministic pushdown automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, and any parity pushdown automaton $\mathcal{A}_{n+1}$, $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ is a $\mathcal{B}(\Sigma_{n+2})$ external winning condition.*
- *There exists a collection of deterministic pushdown automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, and a parity pushdown automaton $\mathcal{A}_{n+1}$ such that $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ is a Π_{n+2}-complete internal winning condition.*

For the internal winning condition $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ on pushdown games, we have the following result.

Theorem 23 *For all $n \geq 0$ the following holds:*

- *For any collection of deterministic pushdown automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, and any parity pushdown automaton $\mathcal{A}_{n+1}$, $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ is a $\mathcal{B}(\Sigma_{n+3})$ internal winning condition for pushdown games.*
- *There exists a collection of deterministic pushdown automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, and a parity pushdown automaton $\mathcal{A}_{n+1}$ such that $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ is a Π_{n+3}-complete internal winning condition for pushdown games.*

Theorem 23 is a consequence of Theorem 20 together with the following lemma.

Lemma 24 *Let $\mathcal{P}$ be a pushdown process with stack alphabet Γ and associated with some pushdown game graph $\mathcal{G}$ equipped with the winning condition $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. Then*

$$\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}} \equiv_W L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})^\sim$$

PROOF. Let $X = \Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ and $Y = L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})^\sim$

Consider the Wadge game $W(X, Y)$. A winning strategy for Bob consists in playing so that if u is the word he has written since the beginning of the play, $u^{\leftarrow_\mathcal{P}}$ equals the stack contents in the last configuration (that is the last letter) written by Alice.

Conversely, consider the Wadge game $W(Y, X)$. The winning strategy for Bob is to write a configuration which stack contents equal to $u^{\hookleftarrow}$, where u is the word already played by Alice. $\square$

4 Decidability

In this section, we explain how to decide the winner in a game played on a finite game graph equipped with a winning condition of the form $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ and in a pushdown game equipped with a winning condition of the form $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$. More precisely, we show the following result.

Theorem 25 *Consider some integer $n \geq 0$ and a collection $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ of pushdown automata where in addition $\mathcal{A}_{n+1}$ is equipped with a parity condition. Then the following holds:*

- *Let $\mathcal{G}$ be a finite game graph. Then, for any vertex v in $\mathcal{G}$, it is decidable whether Eve has a winning strategy from v in the game $\mathbb{G} = (\mathcal{G}, \Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}})$.*
- *Let $\mathcal{G}$ be a pushdown game graph. Then, for any configuration of the form $(q, \bot)$ in $\mathcal{G}$, it is decidable whether Eve has a winning strategy from $(q, \bot)$ in the game $\mathbb{G} = (\mathcal{G}, \Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}})$.*

The proof follows from theorems 26, 27 and 28 that show how to transform a game on a finite game graph into an equivalent pushdown game with a simpler winning condition, and how to transform a pushdown game into an equivalent game on a finite game graph with a winning condition of the same complexity. Thus, several transformations yield a pushdown game with a parity winning condition, a problem known to be decidable [24].

4.1 From a Game on a Finite Game Graph to a Pushdown Game

In this subsection, we consider a finite game graph $\overline{\mathcal{G}} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ with edges labeled by letters on some alphabet A or by the empty word ε, and we set $V = V_{\mathbf{E}} \cup V_{\mathbf{A}}$. We also consider some integer $n \geq 0$, and a collection $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ of pushdown automata, where in addition $\mathcal{A}_{n+1}$ is equipped with a parity condition and such that A is the input alphabet of $\mathcal{A}_1$. Finally, we consider an initial vertex $v_{in} \in V$.

We construct a pushdown game equipped with the winning condition $\Omega^{int}_{\mathcal{A}_2 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ if $n \geq 1$, and with a parity condition if $n = 0$. In addition, we show that there is a position in that game such that it is winning for Eve if and only if v_{in} is winning for Eve in the game $\overline{\mathbb{G}} = (\overline{\mathcal{G}}, \Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}})$.

From $\overline{\mathcal{G}}$ and $\mathcal{A}_1 = \langle Q, \Gamma, A, \bot, q_{in}, \delta \rangle$, we define the following pushdown process $\mathcal{P} = \langle Q \times V, \Gamma, \bot, \Delta \rangle$ where:

- $skip((q', v')) \in \Delta((q, v), \gamma)$ if and only if there is some $a \in A$ such that $(v, a, v') \in E$ and $\delta(q, \gamma, a) = skip(q')$, or $(v, \varepsilon, v') \in E$ and $q = q'$.
- $pop((q', v')) \in \Delta((q, v), \gamma)$ if and only if there is some $a \in A$ such that $(v, a, v') \in E$ and $\delta(q, \gamma, a) = pop(q')$.
- $push((q', v'), \gamma') \in \Delta((q, v), \gamma)$ if and only if there is some $a \in A$ such that $(v, a, v') \in E$ and $\delta(q, \gamma, a) = push(q', \gamma')$.

We consider the partition $Q \times V_{\mathbf{E}} \cup Q \times V_{\mathbf{A}}$ of $Q \times V$, and denote by $\mathcal{G}$ the pushdown game graph induced by $\mathcal{P}$ and the preceding partition. Intuitively, $\mathcal{G}$ encodes on-the-fly computations of $\mathcal{A}_1$ on plays in $\overline{\mathcal{G}}$. Then, we have the following result.

Theorem 26 *If $n \geq 1$, for any vertex $v_{in} \in V$, Eve has a winning strategy in $\overline{\mathbb{G}} = (\overline{\mathcal{G}}, \Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}})$ from v_{in} if and only if she has a winning strategy in $\mathbb{G} = (\mathcal{G}, \Omega^{int}_{\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}})$ from $((v_{in}, q_{in}), \bot)$.*

PROOF.

Assume that Eve has a winning strategy $\overline{f}$ from v_{in} in $\overline{\mathbb{G}}$. We define a strategy f in $\mathbb{G}$ from $((q_{in}, v_{in}), \bot)$. As $\mathcal{G}$ encodes an on-the-fly computation of $\mathcal{A}_1$ on a play in $\overline{\mathcal{G}}$, f will reconstruct a play in $\overline{\mathcal{G}}$ (using some function τ), and use the value of $\overline{f}$ on it, to compute the move to play, and use δ to update the stack and the first component of the control state.

Let λ be a partial play in $\mathbb{G}$ starting from $((q_{in}, v_{in}), \bot)$ and let $\tau(\lambda)$ be inductively defined by:

- If $\lambda = ((q_{in}, v_{in}), \bot)$, then $\tau(\lambda) = v_{in}$.
- If $\lambda = \lambda' \cdot ((q', v'), \sigma')$, then, let $((q, v), \sigma)$ be the last vertex of λ'. Then $\tau(\lambda) = \tau(\lambda') \cdot a \cdot v'$ for some $a \in A \cup \{\varepsilon\}$ such that $(v, a, v') \in E$ and (q', σ') is the successor of (q, σ) by a in $\mathcal{A}_1$. Note that by construction of $\mathcal{P}$, $\tau(\lambda)$ is always defined (and may not be unique).

Now, for any partial play λ ending in some configuration $((q, v), \sigma)$, we set $f(\lambda) = ((q', v'), \sigma')$ where $(v, a, v') = \overline{f}(\tau(\lambda))$ for some $a \in A \cup \{\varepsilon\}$, and where (q', σ') is the successor by a of (q, σ) in $\mathcal{A}_1$.

Now, it is easily seen that any partial play λ in $\mathcal{G}$ starting from $((q_{in}, v_{in}), \bot)$ where Eve respects f is such that:

(1) $\tau(\lambda)$ is a play in $\overline{\mathcal{G}}$ starting from v_{in} where Eve respects $\overline{f}$.

(2) The run of $\mathcal{A}_1$ on $\mathrm{Lab}(\tau(\lambda))$ ends in the configuration (q, σ) where the last vertex of λ is of the form $((q, v), \sigma)$ for some $v \in V$.

Now, we conclude that a play λ in $\mathcal{G}$, where Eve respects f, is winning for her. If the play is finite then the first point allows us to conclude that the player who cannot move is Adam. If λ is infinite, $\tau(\lambda)$ is won by Eve, and therefore the stack of $\mathcal{A}_1$ when reading $\mathrm{Lab}(\tau(\lambda))$ is unbounded and has its limits in $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Therefore, the stack in λ is unbounded and its limit is in $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Thus $\lambda \in \Omega^{int}_{\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$.

Conversely, assume that Eve has a winning strategy f from $((q_{in}, v_{in}), \bot)$ in $\mathbb{G}$. Let $\overline{\lambda}$ be a partial play in $\overline{\mathbb{G}}$ starting from v_{in} and let $\pi(\overline{\lambda})$ be inductively defined by:

- If $\overline{\lambda} = v_{in}$, then $\pi(\overline{\lambda}) = ((q_{in}, v_{in}), \bot)$.
- If $\overline{\lambda} = \overline{\lambda}' \cdot a \cdot v'$, then, let $((q, v), \sigma)$ be the last vertex of $\pi(\overline{\lambda}')$. Then $\pi(\overline{\lambda}) = \pi(\overline{\lambda}') \cdot ((q', v'), \sigma')$ where (q', σ') is the successor by a of (q, σ) in $\mathcal{A}_1$.

Now, for any partial play $\overline{\lambda}$ ending in some configuration v, we set $\overline{f}(\overline{\lambda}) = (v, a, v')$ where $((q', v'), \sigma') = f(\pi(\overline{\lambda}))$ and $a \in A \cup \{\varepsilon\}$ is such that (q', σ') is the successor in $\mathcal{A}_1$ by a of (q, σ), where $((q, v), \sigma)$ designates the last configuration of $\pi(\overline{\lambda})$. Note that, by definition of λ, such an a always exists (and may not be unique).

Now, it is easily seen that any partial play $\overline{\lambda}$ in $\overline{\mathcal{G}}$ starting from v_{in} where Eve respects $\overline{f}$ is such that:

(1) $\pi(\lambda)$ is a play in $\mathcal{G}$ starting from $((v_{in}, q_{in}), \bot)$ where Eve respects f.
(2) The run of $\mathcal{A}_1$ on $\mathrm{Lab}(\overline{\lambda})$ ends in the configuration (q, σ) where the last vertex of $\pi(\overline{\lambda})$ is of the form $((q, v), \sigma)$, for some $v \in V$.

Now, we easily conclude that a play $\overline{\lambda}$ in $\overline{\mathcal{G}}$, where Eve respects $\overline{f}$, is winning for her. If the play is finite then the first point allows to conclude that the player who cannot move is Adam. If $\overline{\lambda}$ is infinite, $\pi(\overline{\lambda})$ is won by Eve, and therefore the stack is unbounded in $\pi(\overline{\lambda})$ and its limits is in $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Therefore, the stack of $\mathcal{A}_1$ when reading $\mathrm{Lab}(\overline{\lambda})$ is unbounded and its limit is in $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Thus $\overline{\lambda}$ is winning for the external condition $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. □

For the case where $n = 0$, $\mathcal{A}_1$ is equipped with a coloring function $col : Q \to C$. Therefore, we extend col into a function $col : Q \times V \to C$ by setting $col(q, v) = col(q)$. We denote by $\mathbb{G}$ the pushdown parity game played on $\mathcal{G}$ and induced by col. Then, using the same techniques as for Theorem 26, we prove the following result.

Theorem 27 *For any vertex $v_{in} \in V$, Eve has a winning strategy in $\overline{\mathbb{G}} = (\overline{\mathcal{G}}, \Omega^{ext}_{\mathcal{A}_1})$ from v_{in} if and only if she has a winning strategy in the parity pushdown game $\mathbb{G}$ from $((v_{in}, q_{in}), \bot)$.*

4.2 From a Pushdown Game to a Game on a Finite Game Graph

In this section, we consider a pushdown process $\mathcal{P} = \langle Q, \Gamma, \bot, \Delta \rangle$ together with a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q. The associated pushdown game graph is denoted by $\mathcal{G}$. We also consider some integer $n \geq 0$ and a collection $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ of pushdown automata where in addition $\mathcal{A}_{n+1}$ is equipped with a parity condition and such that Γ is the input alphabet of $\mathcal{A}_1$. Finally, we set $\mathbb{G} = (\mathcal{G}, \Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}})$ and we consider an initial configuration $(p_{in}, \bot)$ in $\mathcal{G}$.

We now construct a *finite* game graph $\overline{\mathcal{G}}$ with the external winning condition $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$, such that Eve has a winning strategy in $\mathcal{G}$ from $(p_{in}, \bot)$ if and only if she has a winning strategy in $\overline{\mathcal{G}}$ from some special vertex. Intuitively, in $\overline{\mathcal{G}}$, we keep track only of the control state and the symbol on the top of the stack. The interesting aspect of the game is when it is in a control state p with top-of-stack γ, and the player owning p wants to push a letter γ' onto the stack. Consider the set of all (finite) continuations of the play that will end with popping this γ' symbol from the stack. We require Eve to declare the set of all states the game can be in after the popping of γ' along these plays. Note that since it's a game, Eve does not have complete control, hence she cannot give one exact state, but can only give the set of possible states the game could be in at the corresponding pop. Let this set be R.

Adam now has two choices. He can either continue the game by pushing γ' onto the stack and updating the state (we call this a *pursue* move), or he can pick some state $p'' \in R$ and continue from that state, leaving γ on the top of the stack (we call this a *jump* move). If he makes a pursue move, then he remembers R, and makes sure that if there is a pop-transition on γ' later in the play, then the resulting state is indeed in R (if it is not, Eve would lose the game). If along a play such a pop-transition on γ' is indeed met, and the resulting state is in R, then the play stops right there and Eve is declared the winner.

Now consider an infinite play in this game, that is a play that never simulates a pop-transition. In such a play, a pursue move corresponds to a new letter pushed forever on the stack. Therefore, if there are only finitely many pursue moves, the stack is not strictly unbounded (some stack level is infinitely often visited), and therefore the play will be lost for Eve. If it goes infinitely often through pursue moves, the stack is strictly unbounded and its limit is obtained by considering the word which letters are the top stack symbols just before

pursue moves. Hence, the winning condition becomes external, if we label the pursue move by the top stack symbol in the configuration just before the pursue move.

Let us now describe the construction more precisely. The structure of the finite game graph $\overline{\mathcal{G}}$ is depicted in Figure 2.

The main vertices of $\overline{\mathcal{G}}$ are tuples in $Q \times \Gamma \times 2^Q$. A vertex (p, γ, R) belongs to Eve if and only if $p \in Q_{\mathbf{E}}$. Intuitively, a vertex (p, γ, R) denotes that p is the current state, γ is the symbol on the top of the stack and R is the current commitment Eve has made, i.e. Eve has previously claimed that if there is a pop-γ transition, then the state will be in R. The starting vertex is $(p_{in}, \bot, \emptyset)$ (the bottom-of-stack symbol will never be popped, and therefore the third component is $\emptyset$).

In order to simulate an internal-transition $skip(p') \in \Delta(p, \gamma)$, we have edges in $\overline{\mathcal{G}}$ of the form $((p, \gamma, R), \varepsilon, (p', \gamma, R))$, for all $R \subseteq Q$.

Pop-transitions are not simulated. From any vertex (p, γ, R), we have a transition to tt if $\exists r \in R$ such that $pop(r) \in \Delta(p, \gamma)$ (showing that R was correctly defined with respect to this transition). Also, we have a transition to ff, if $\exists r \notin R$ such that $pop(r) \in \Delta(p, \gamma)$ (showing that R was not correctly defined with respect to this transition).

The simulation of a push-transition goes in several steps. Let (p, γ, R) be a vertex. The player owning p first picks a particular push-transition $push(p', \gamma') \in \Delta(p, \gamma)$ by moving (through an edge labeled by ε) to the vertex $(p, \gamma, R, p', \gamma')$, which belongs to Eve. Then Eve proposes a set $R' \subseteq Q$ containing the states that she claims to be reached if γ' gets eventually popped. She does this by moving (through an edge labeled by ε) to the vertex $(p, \gamma, R, p', \gamma', R')$, which belongs to Adam.

Now, Adam has two kinds of choices. He can do a *jump* move by picking a state $p'' \in R'$, and move to the vertex (p'', γ, R) through an edge labeled by ε. Or he can do a *pursue* move by moving to the vertex (p', γ', R') through an edge labeled by γ. Note that these last edges are the only ones that are not labeled by ε.

Now we have the following result.

Theorem 28 *Eve has a winning strategy from $(p_{in}, \bot)$ in $\mathbb{G}$ if and only if she has a winning strategy from $(p_{in}, \bot, \emptyset)$ in $\overline{\mathbb{G}} = (\overline{\mathcal{G}}, \Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}})$.*

Before starting the proof, we introduce two notations. Let λ be some play in $\mathcal{G}$, we set $Steps_\lambda = \{n \in \mathbb{N} \mid \forall m \geq n,\ |\lambda(m)| \geq |\lambda(n)|\}$, where $|\lambda(n)|$ is the size of the stack when being in the configuration $\lambda(n)$. Therefore $Steps_\lambda$ is

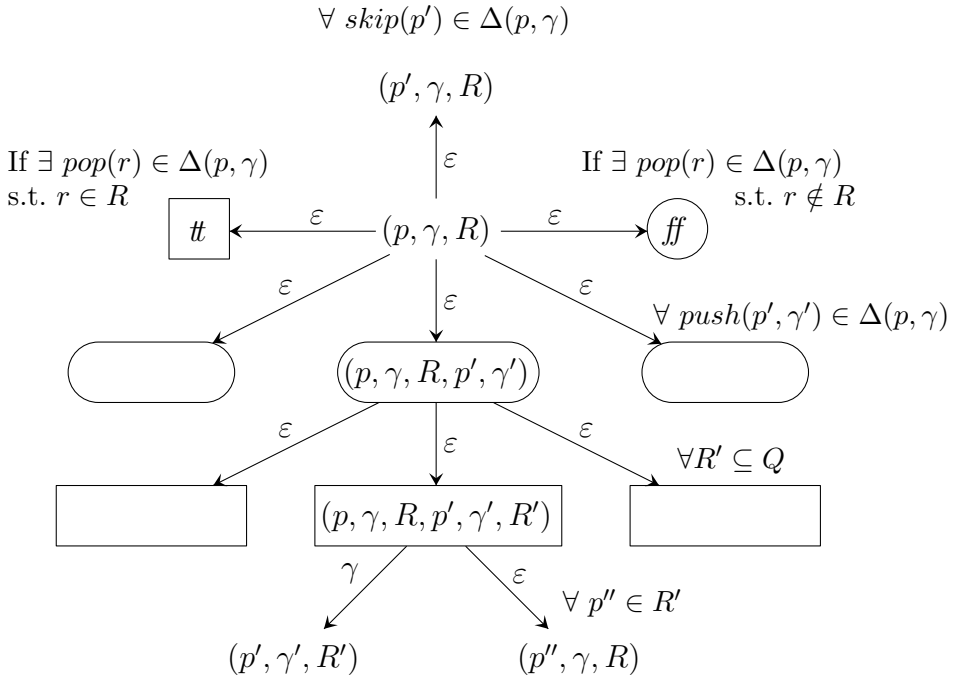


Fig. 2. Local structure of $\overline{\mathcal{G}}$: oval vertices belong to Eve, square for Adam.

the set of indexes corresponding to positions where the stack height will not decrease later on the play.

Let $\overline{\lambda}$ be some play in $\overline{\mathcal{G}}$, we set $Steps_{\overline{\lambda}} = \{n \in \mathbb{N} \mid \overline{\lambda}(n) = (p,\gamma,R),\ \text{for some } p \in Q,\ \gamma \in \Gamma,\ R \subseteq Q\}$. Therefore $Steps_{\overline{\lambda}}$ is the set of indexes corresponding to positions where the play is in one of the main vertices of $\overline{\mathcal{G}}$.

From $\mathbb{G}$ to $\overline{\mathbb{G}}$.

Assume that $(p_{in},\bot)$ is winning for Eve in the game $\mathbb{G}$. This means that Eve has a winning strategy $f : V^*V_{\mathbf{E}} \to V$ from $(p_{in},\bot)$.

We define, using f, a strategy $\overline{f}$ for Eve in $\overline{\mathbb{G}}$ from $(p_{in},\bot,\emptyset)$. For this we inductively construct a play λ in $\mathbb{G}$ and consider the prefix of λ already constructed to determine how to play in $\overline{\mathbb{G}}$ at any time: $\overline{f}$ is defined from f and from the prefix of λ already constructed.

Let us describe λ and $\overline{f}$:

(1) At the beginning $\lambda = (p_{in},\bot)$.
(2) Assume that the play is in some configuration (p,γ,R). If $p \in Q_{\mathbf{E}}$, Eve considers the move given in $\mathbb{G}$ by f when having played λ (for instance at the beginning from $(p_{in},\bot)$). In other words, she considers the value of $f(\lambda)$. If it is a skip-transition $skip(p')$, she moves to (p',γ,R). If it is

a pop-transition then she goes to $\#\!\!\#$ (Proposition 29 will show that such a move is always possible in that case). Otherwise she goes to the vertex $(p, \gamma, R, p', \gamma')$, where the push-transition was $push(p', \gamma')$.

If $p \in Q_{\mathbf{A}}$, Adam goes either to $\not\#\not\#$ (Proposition 29 will show that such a move is impossible), to $\#\!\!\#$, to some vertex (p', γ, R) or to some vertex $(p, \gamma, R, p', \gamma')$.

(3) From a vertex $(p, \gamma, R, p', \gamma')$, Eve considers the set of all (finite) extensions of $\lambda \cdot (p', \gamma'\gamma\sigma)$ (where $(p, \gamma\sigma)$ was the last configuration in λ) in $\mathbb{G}$, where she respects f, and that end with popping γ' from the stack. She moves to $(p, \gamma, R, p', \gamma', R')$, where R' is the set of all states the game can be in after popping γ' along these plays.

(4) Assume Adam goes from $(p, \gamma, R, p', \gamma', R')$ to (p'', γ, R), by playing a jump move. If $(p, \gamma\sigma)$ is the last configuration in λ, then Eve completes λ by adding $(p', \gamma'\gamma\sigma)$ followed by a sequence of moves respecting f in $\mathbb{G}$ that ends in $(p'', \gamma\sigma)$. Then she goes to step (2).

(5) Assume Adam goes from $(p, \gamma, R, p', \gamma', R')$ to (p', γ', R') by playing a pursue move. If $(p, \gamma\sigma)$ is the last configuration in λ, then Eve completes λ by adding $(p', \gamma'\gamma\sigma)$. Then she goes to step (2).

Therefore with any partial (resp. infinite) play $\overline{\lambda}$ in $\overline{\mathbb{G}}$ is associated a partial (resp. infinite) play λ in $\mathbb{G}$.

From the definition of f, it follows that for any partial play $\overline{\lambda}$ in $\overline{\mathbb{G}}$, where Eve respects $\overline{f}$, the corresponding partial play λ is a valid partial play in $\mathbb{G}$ where Eve follows her winning strategy f. The same holds for infinite plays. Therefore, if λ is infinite, it is a winning play for Eve.

In addition, we have the following proposition, which is a direct consequence of how $\overline{f}$ is defined.

Proposition 29 *Consider a partial play $\overline{\lambda}$ in $\overline{\mathbb{G}}$ where Eve respects $\overline{f}$ and that ends in a vertex (p, γ, R). Then, the associated partial play λ (constructed by $\overline{f}$) ends in some vertex $(p, \gamma\sigma)$ for some $\sigma \in St$. Moreover, for any continuation of λ where Eve respects f, if γ is eventually popped and leads to some configuration (r, σ), then $r \in R$.*

In particular, the above proposition implies that $\overline{\lambda}$ does not eventually reach some vertex $\not\#\not\#$ (such a move would be done by Adam and would correspond to a pop-transition from λ that contradicts Proposition 29) and that the moves to $\#\!\!\#$ given by $\overline{f}$ are always possible. Therefore, finite plays in $\overline{\mathbb{G}}$ are won by Eve.

Now, consider some infinite play $\overline{\lambda}$ in $\overline{\mathbb{G}}$ starting from $(p_{in}, \bot, \emptyset)$ where Eve respects $\overline{f}$. Let λ be the infinite play constructed by $\overline{f}$ while playing $\overline{\lambda}$. λ is won by Eve as it is a play where she respects her winning strategy f. Moreover, we have the following straightforward result.

Lemma 30 *Let* $Steps_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$ *and let* $Steps_{\overline{\lambda}} = \{m_0 < m_1 < m_2 < \cdots\}$. *Then for any index* $i \geq 0$, *one has the following:*

- $\lambda(n_i) = (p, \gamma\sigma)$ *for some* $p \in Q$, $\gamma \in \Gamma$ *and* $\sigma \in St$ *if and only if* $\overline{\lambda}(m_i) = (p, \gamma, R)$ *for some* $R \subseteq Q$.
- $|\lambda(n_i)| = |\lambda(n_{i+1})|$ *if and only if in the factor* $\overline{\lambda}(m_i) \cdots \overline{\lambda}(m_{i+1})$ *of* $\overline{\lambda}$, *all the edges are labeled by* ε.
- $|\lambda(n_i)| + 1 = |\lambda(n_{i+1})|$ *if and only if in the factor* $\overline{\lambda}(m_i) \cdots \overline{\lambda}(m_{i+1})$ *of* $\overline{\lambda}$, *there is exactly one edge which is not labeled by* ε. *In addition, it is labeled by* γ *where* γ *is such that* $\lambda(n_i) = (p, \gamma\sigma)$ *for some* $p \in Q$ *and* $\sigma \in St$.

Now, as λ is an infinite play starting from $(p_{in}, \bot)$, where Eve respects her winning strategy f, it implies that the stack of $\mathcal{P}$ is strictly unbounded in λ and therefore $\mathrm{StLim}(\lambda) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Therefore, using Lemma 30, we conclude that $\mathrm{Lab}(\overline{\lambda}) = \mathrm{StLim}(\lambda)$ and therefore, $\mathrm{Lab}(\overline{\lambda}) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}) = \Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$, which exactly means that $\overline{\lambda}$ is winning for Eve.

From $\overline{\mathbb{G}}$ to $\mathbb{G}$.

Assume that $(p_{in}, \bot, \emptyset)$ is winning for Eve in $\overline{\mathbb{G}}$ and let $\overline{f}$ be a winning strategy for Eve in $\overline{\mathbb{G}}$ from $(p_{in}, \bot, \emptyset)$. From $\overline{f}$, we define a winning strategy f in $\mathbb{G}$ from $(p_{in}, \bot)$.

The strategy f uses a stack Υ containing vertices of $\overline{\mathcal{G}}$. At the beginning, Υ only contains $(p_{in}, \bot, \emptyset)$. By $top(\Upsilon)$ we denote the top symbol of Υ. We will have $top(\Upsilon) = (p, \gamma, R)$ if and only if the current partial play λ in $\mathbb{G}$ ends in some configuration $(p, \gamma\sigma)$ for some $\sigma \in St$. In addition, R is such that, if Eve respects f, and if γ is eventually popped, it leads to some state $r \in R$. By $StCont(\Upsilon)$ we denote the word obtain by reading Υ from bottom to top. $StCont(\Upsilon)$ will contain a play in $\overline{\mathbb{G}}$ starting from $(p_{in}, \bot, \emptyset)$, and where Eve respects her winning strategy $\overline{f}$.

In order to describe f, let us assume that we are in some configuration $(p, \gamma\sigma)$ with $top(\Upsilon) = (p, \gamma, R)$. First we describe how Eve plays if she is the one that has to move, and then we explain how Υ is updated.

- **Choice of the move:** From some configuration $(p, \gamma\sigma)$ where $p \in Q_{\mathbf{E}}$, Eve considers the value of $\overline{f}(StCont(\Upsilon))$. If the move given by $\overline{f}(StCont(\Upsilon))$ is to some vertex (p', γ, R), then, in $\mathbb{G}$, Eve plays the internal-transition $skip(p') \in \Delta(p, \gamma)$.

  If it is a move to $\#$ then Eve plays some pop-transition $pop(r) \in \Delta(p, \gamma)$ for some $r \in R$ (Lemma 31 will show that it is always possible).

  If the move given by $\overline{f}(StCont(\Upsilon))$ is to some vertex $(p, \gamma, R, p', \gamma')$, then, in $\mathbb{G}$, Eve plays the push-transition $push(p', \gamma') \in \Delta(p, \gamma)$.

- **Update of Υ:** If the move (made by whoever) from $(p, \gamma\sigma)$ is to move to a configuration $(p', \gamma\sigma)$, Eve updates Υ by pushing the transition $((p, \gamma, R), \varepsilon, (p', \gamma, R))$ followed by the vertex (p', γ, R).

  If the move (made by whoever) from $(p, \gamma\sigma)$ is to pop and reach a configuration (r, σ), Eve updates Υ by popping the top symbols until finding a transition labeled by some letter $\gamma' \neq \varepsilon$. Let $((q', \gamma', R', q, \gamma, R), \gamma', (q, \gamma, R))$ be this top symbol. She removes it and pushes the edge $((q', \gamma', R', q, \gamma, R), \varepsilon, (r, \gamma', R'))$ followed by the vertex (r, γ', R'). This update is illustrated in Figure 3.

  If the move (made by whoever) from $(p, \gamma\sigma)$ is to push γ' and reach some configuration $(p', \gamma'\gamma\sigma)$, then Eve pushes on Υ's top the edge $((p, \gamma, R), \varepsilon, (p, \gamma, R, p', \gamma'))$ followed by the vertex $(p, \gamma, R, p', \gamma')$. Then she pushes the value $((p, \gamma, R, p', \gamma', R'), \varepsilon, (p, \gamma, R, p', \gamma', R'))$ of $\overline{f}(StCont(\Upsilon))$ followed by $(p, \gamma, R, p', \gamma', R')$ on Υ's top. Finally, she pushes the transition $((p, \gamma, R, p', \gamma', R'), \gamma, (p', \gamma', R'))$ followed by (p', γ', R') on Υ's top.

We then have the following lemma.

Lemma 31 *Let λ be a partial play starting from $(p_{in}, \bot)$ in $\mathbb{G}$ where Eve follows her strategy f and that ends in a configuration $(p, \gamma\sigma)$. Then, the following holds:*

(1) $top(\Upsilon) = (p, \gamma, R)$ for some $R \subseteq Q$.
(2) $StCont(\Upsilon)$ is a partial play in $\overline{\mathbb{G}}$ that starts in $(p_{in}, \bot, \emptyset)$, ends in (p, γ, R) and in which Eve respects $\overline{f}$.
(3) If the next move is a pop-γ-transition, then it leads to some configuration (r, σ) with $r \in R$.

PROOF.

The proof goes by induction on the length of the play.

First we show that the third point is a direct consequence of the two other points. Assume that from $(p, \gamma\sigma)$ a pop-transition $pop(r) \in \Delta(p, \gamma)$ is applied. Therefore, there is an edge in $\overline{\mathcal{G}}$ from (p, γ, R) to $\#$ or ff depending whether $r \in R$. If $p \in Q_{\mathbf{E}}$ then, due to how f is defined, the edge is to $\#$ and thus we have the result. If $p \in Q_{\mathbf{A}}$ then, if $r \notin R$, Adam could move in $\overline{\mathcal{G}}$ from (p, γ, R) to ff: using the second point, $StCont(\Upsilon)$ is a partial play that ends in (p, γ, R) and where Eve respects her winning strategy $\overline{f}$. Thus, it cannot be extended by a move of Adam into a loosing play for Eve.

Now assume that the result holds for some play λ, and let us show that it also holds for some play λ' obtained from λ by applying some transition. We have two cases, depending on the kind of transition is applied :

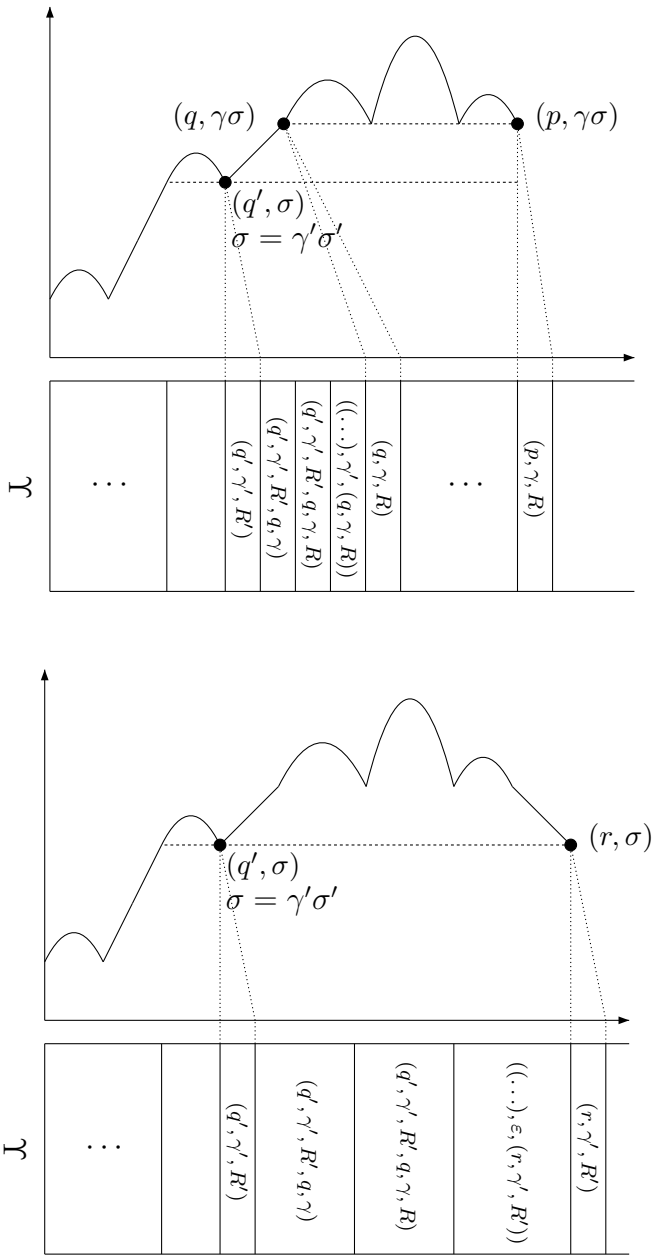


Fig. 3. Updating Υ after popping

- λ' is obtained from λ by playing some internal-transition or some push-transition. The first two points are immediate using the induction hypothesis and considering the way Υ is updated.
- λ' is obtained from λ by applying some pop-γ-transition. Then let $(p, \gamma\sigma)$ be the last configuration of λ and let R be the third component of $top(\Upsilon)$ in $(p, \gamma\sigma)$. Then by induction hypothesis, and using the third point, $\lambda' = \lambda \cdot (r, \sigma)$ for some $r \in R$. In addition, it is easily seen that, if $n = |\sigma|$, there are n edges with labels different from ε, and that $StCont(\Upsilon) = \overline{\lambda}_1 \cdot (p', \gamma', R', q, \gamma, R) \cdot ((p', \gamma', R', q, \gamma, R), \gamma', (q, \gamma, R)) \cdot \overline{\lambda}_2 \cdot (p, \gamma, R)$ for some

$p', q \in Q$, $R' \subseteq Q$, $\gamma' \in \Gamma$, and for some partial plays $\overline{\lambda}_1, \overline{\lambda}_2$ of $\overline{\mathcal{G}}$. In addition, all vertices in $\overline{\lambda}_2$ have R as third component and $\overline{\lambda}_2$ only contains edges labeled by ε. Now if we denote by Υ' the updated value of Υ, by definition of how the stack Υ is updated, $StCont(\Upsilon') = \overline{\lambda}_1 \cdot (p', \gamma', R', q, \gamma, R) \cdot ((p', \gamma', R', q, \gamma, R), \varepsilon, (r, \gamma', R')) \cdot (r, \gamma', R')$. By induction hypothesis on $StCont(\Upsilon)$ and as $r \in R$, $StCont(\Upsilon')$ is a valid play in $\overline{\mathbb{G}}$ where Eve respects $\overline{f}$. This shows that the second point holds. The first points is a direct consequence of the second one. $\square$

By induction, one easily has the following proposition.

Proposition 32 *Let λ be a partial play starting from $(q_{in}, \bot)$ in $\mathbb{G}$ where Eve follows her strategy f, and that ends in a configuration $(p, \gamma\sigma)$ where $top(\Upsilon) = (p, \gamma, R)$ when being in $(p, \gamma\sigma)$. Let $\overline{\lambda}$ be equal to $StCont(\Upsilon)$. Let $Steps_\lambda = \{n_0 < n_1 < n_2 < \cdots < n_h\}$ and let $Steps_{\overline{\lambda}} = \{m_0 < m_1 < m_2 < \cdots < m_k\}$. Then the following holds:*

- *$h = k$.*
- *For every $i = 0, \ldots, h$, $\lambda(n_i)$ is of the form $(p, \gamma\sigma)$ for some $p \in Q$, $\gamma \in \Gamma$ and $\sigma \in St$ if and only if $\overline{\lambda}(m_i)$ is of the form (p, γ, R) for some $R \subseteq Q$ and $\sigma = \mathrm{Lab}(\overline{\lambda}{\restriction_{m_i}})$.*

Now consider an infinite play λ in $\mathbb{G}$ where Eve respects f. Then, either some level is infinitely repeated or not. For every integer $i \geq 0$, let $\overline{\lambda}_i = StCont(\Upsilon_i)$, where Υ_i is the (strategy) stack in $\lambda(i)$. Then, in both cases, for all $k \geq 0$ there is some $j \geq 0$ such that $\overline{\lambda}_i{\restriction_k} = \overline{\lambda}_j{\restriction_k}$ for all $i \geq j$. Let $\overline{\lambda}$ be the infinite word such that for all $k \geq 0$, $\overline{\lambda}{\restriction_k}$ is the limit of $(\overline{\lambda}_i{\restriction_k})_{i \geq 0}$. Then, from Lemma 31, it follows that $\overline{\lambda}$ is a play in $\overline{\mathbb{G}}$ starting from $(p_{in}, \bot, \emptyset)$ where Eve respects $\overline{f}$. Therefore $\mathrm{Lab}(\overline{\lambda}) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ and is thus infinite. Using Proposition 32, we conclude that the stack is strictly unbounded in λ and that $\mathrm{StLim}(\lambda) = \mathrm{Lab}(\overline{\lambda})$. Therefore $\mathrm{StLim}(\lambda) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ and thus λ is winning for Eve. $\square$

4.3 The Special Case of Strict Unboundedness Pushdown Games

In this section, we consider the special case of strict unboundedness pushdown games. In [4], the following internal winning condition on pushdown game graphs was considered: Eve wins an infinite play if and only if some configuration is infinitely often visited. It is easily seen that the corresponding winning condition for Adam is the strict unboundedness winning condition.

Since the strict unboundedness winning condition is the winning condition $\Omega^{int}_{\mathcal{A}}$ for any automaton $\mathcal{A}$ recognizing the ω-regular language Γ^ω (where Γ is

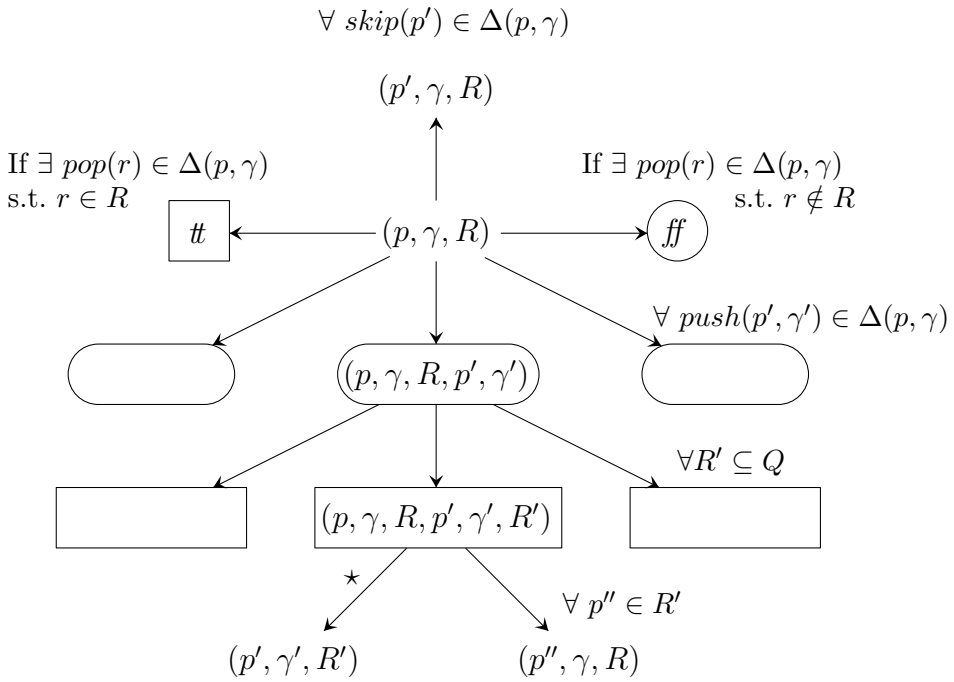


Fig. 4. Local structure of $\overline{\mathcal{G}}$ for strict unboundedness: final edges are marked by $\star$.

the stack alphabet of the pushdown graph), we conclude that the preceding constructions induce the decidability for the strict unboundedness pushdown game (and for the games considered in [4]).

More precisely, let $\mathcal{P} = \langle Q, \Gamma, \bot, \Delta \rangle$ be a pushdown process with a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q. We denote by $\mathcal{G}$ the associated pushdown game graph, and by $\mathbb{G}_{ubd}$ the associated strict unboundedness pushdown game. Then, the *equivalent* game, played on a finite graph described as above, is equipped with the winning condition $\Omega^{ext}_{\mathcal{A}}$. Therefore, this last game is a Büchi game, as Eve wins a play if and only if infinitely many edges not labeled by ε are visited. Therefore, we have the following special case of Theorem 28.

Theorem 33 *Eve has a winning strategy from a configuration $(p_{in}, \bot)$ in $\mathbb{G}$ if and only if she has a winning strategy from $(p_{in}, \bot, \emptyset)$ in the Büchi game $\overline{\mathbb{G}}$ played on the finite game graph $\overline{\mathcal{G}}$ (with edges marked final) depicted in Figure 4.*

Therefore, we have the following corollary.

Corollary 34 *Deciding the winner in a strict unboundedness pushdown game (respectively in a pushdown game equipped with the winning condition of [4]) can be achieved in* Dexptime.

5 Complexity

5.1 Main Results

We first start with some definitions.

Definition 35 *For any $h, N \geq 0$, $tow(h, N)$ is defined inductively by:*

- $tow(0, N) = N$.
- $tow(h, N) = 2^{tow(h-1,N)}$ *for* $h \geq 1$.

For instance, $tow(3, N) = 2^{2^{2^N}}$.

Definition 36 (h-DEXPTIME) *Let consider a problem P and a deterministic Turing machine deciding in $\mathcal{O}(tow(h, N))$ steps whether some instance of the problem P is true, where N is polynomial in the size of the instance. Then the problem P belongs to the class h-DEXPTIME.*

We have the following upper bounds.

Proposition 37 *Let consider an integer $k \geq 0$ and a collection $\mathcal{A}_1, \ldots, \mathcal{A}_k$, $\mathcal{A}_{k+1}$ of pushdown automata, where in addition $\mathcal{A}_{k+1}$ is equipped with a parity condition.*

Let $\mathbb{G}$ be a pushdown game equipped with the winning condition $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$. Deciding the winner in such a game is a $(k+2)$-DEXPTIME problem.

Let $\mathbb{G}$ be a game on a finite game graph equipped with the winning condition $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$. Deciding the winner in such a game is a $(k+1)$-DEXPTIME problem.

PROOF. First note that the transformation described in section 4.1 is polynomial, and that the transformation described in section 4.2 is exponential. Moreover, recall that deciding the winner in a parity pushdown game costs exponential time [24]. Then the proof is immediate by induction on k. □

Concerning the lower bound, we have the following result which is proved in section 5.2.

Proposition 38 *The problem of deciding the winner in a game played on a finite game graph and equipped with a winning condition of the form $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$, with $k \geq 0$, is a $(k+1)$-DEXPTIME hard problem.*

From Proposition 38 and from the polynomial reduction described in section 4.1, we obtain the following corollary.

Corollary 39 *The problem of deciding the winner in a pushdown game equipped with a winning condition of the form* $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$*, with* $k \geq 0$*, is a* $(k+2)$-DEXPTIME *hard problem.*

Recall that a problem is *elementary* if it is a k-DEXPTIME problem for some $k \geq 0$. An *elementary hard* problem is thus a problem which is k-DEXPTIME hard for all $k \geq 0$.

Propositions 37 and 38 and Corollary 39 leads to the following result.

Theorem 40 *The complexity of deciding the winner in a pushdown game (resp. a game on a finite game graph) equipped with a winning condition of the form* $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$ *(resp.* $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$*) is non-elementary and is elementary-hard.*

5.2 Proof of Proposition 38

5.2.1 Presentation of the Proof

The proof starts with the basic case $k = 0$. Then, it focuses on the case $k = 1$ that gives some insight for the general case. The basic idea of the proof is to simulate an alternating Turing machine using $tow(k, N)$ space for some $k \geq 1$ and some polynomial N in the size of the input. Such a Turing machine is equivalent to a deterministic Turing machine using $tow(k+1, N)$ time [16] which completes the proof.

From now on, a configuration of a Turing machine is described by a word upv, where uv is the contents of the tape, p is the control state and the head of the Turing machine is just after u.

Note that in all the proof $\mathcal{A}_{k+1}$ will not be equipped with a parity acceptance condition but with a weaker one, namely a reachability condition.

5.2.2 Basic Case: $k = 0$

This case is a slightly modified version of the proof showing that deciding the winner in a reachability pushdown game is DEXPTIME hard [24].

5.2.3 Basic Case: $k = 1$

We start by proving the hardness for the special case of a game $\mathbb{G}$ played on a finite game graph $\mathcal{G}$ equipped with a winning condition $\Omega^{ext}_{\mathcal{A}_1 \triangleright \mathcal{A}_2}$. We explain how such a game can be used to simulate an alternating Turing machine M of exponential space 2^N, where N is polynomial in the size of M. More precisely, $\mathbb{G}$ is constructed so that Eve has a winning strategy if and only if M accepts from a blank tape. A play in $\mathbb{G}$ is the description by Eve and Adam of a run of the Turing machine M, that is a sequence of configurations. For this the players go through labeled-edges. In what follows, we say that a player writes a instead of saying that he goes through an edge labeled by a.

A configuration $a_0 a_1 \dots a_{2^N-1}$ of M will be represented by the following word:

$$(\#a_0a_1)(n_0, \widetilde{n}_0)^{2^N}(a_0a_1a_2)(n_1, \widetilde{n}_1)^{2^N} \dots (a_{2^N-2}a_{2^N-1}\#)(n_{2^N-1}, \widetilde{n}_{2^N-1})^{2^N}$$

where n_i is the binary representation of i on N bits with the most significant bit on the left and $\widetilde{n}_i$ is the binary representation of i with the most significant bit on the right. For instance, if $N = 4$, $n_{10} = 1010$ and $\widetilde{n}_{10} = 0101$. By $(n_i, \widetilde{n}_i)$ we mean $(n_i^0, n_i^{N-1})(n_i^1, n_i^{N-2}) \dots (n_i^{N-1}, n_i^0)$ where $n_i = n_i^0 n_i^1 \dots n_i^{N-1}$, *e.g.* $(n_{10}, \widetilde{n}_{10}) = (1,0)(0,1)(1,0)(0,1)$. For a word u and an integer $k \geq 0$, u^k is the word obtained by concatenating k copies of u (*e.g.* $u^4 = uuuu$). The sequence $(n_i, \widetilde{n}_i)$ is called a *counter*. The reason why we copy 2^N times the same counter will appear later when we perform an exhaustive search of a counter in a stack.

A play goes as follows: Eve describes (going through labeled-edges) the initial configuration (that is, the initial state i followed by 2^N-1 blank symbols) then, depending on whether the control state in the initial configuration is existential or universal, Eve (existential state) or Adam (universal state), chooses a move, that is, goes through an edge whose label is the description of a transition of the Turing machine. Then, Eve describes the resulting configuration and so on. If some accepting configuration is eventually reached then the play goes in a special vertex where it loops forever through an edge labeled by a special symbol $\heartsuit$.

To prevent Eve from cheating in the description of the configurations, there are 5 special symbols $!_c, \widetilde{!_c}, !_v, \widetilde{!_v}$ and ? (c stands for copy and v for value). Adam can write them to contest the validity of the last symbol written by Eve. They can be used in the following cases:

- The j-th pair of bits of a counter $(n_i, \widetilde{n}_i)$ is false. If it is the first copy of the counter, the value of the counter should be equal to the value of the preceding counter plus 1. Therefore, Adam writes $!_v$ (if the mistake is for n_i) or $\widetilde{!_v}$ (otherwise) just after the wrong pair of bits. If it is not the first

copy, the error concerns the duplication. In that case, Adam writes $!_c$ (if the mistake is for n_i) or $\widetilde{!_c}$ (otherwise) just after the wrong pair of bits.
- The value of a tuple $(a_{i-1}a_ia_{i+1})$ is incorrect. In that case, Adam writes ? just after it.

Once a contesting $!_c$, $\widetilde{!_c}$, $!_v$ or $\widetilde{!_v}$ has been made, the play goes in a special vertex where it loops forever through an edge labeled by a special symbol ♠. In the case of a contesting ?, the value of the next counter is written (and can be contested), and then the play reaches the aforementioned special vertex and loops forever through an edge labeled by ♠.

The alternation between both players (that allows Adam to contest any symbol written by Eve), the memory to decide whether the control state is existential or universal, and the memory of a contesting or a final configuration (to write ♠ or ♡) are encoded in the vertices of $\mathcal{G}$. It is also the same to change of component of $\mathcal{G}$ when $(n_{2^N-1}, \widetilde{n}_{2^N-1})$ has been written or to force Eve to start describing the empty tape configuration and to write $(n_0, \widetilde{n}_0)$ after the first tuple of a configuration. Moreover, the vertices are also used to force Eve to describe an overlapping sequence of tuples of letters. More precisely, if she has just written the tuple (a_{i-1}, a_i, a_{i+1}), then she will be in some vertex where she will be forced to write as next tuple (a_i, a_{i+1}, a_{i+2}) for some letter a_{i+2}.

Let us now describe how the validity of a contesting is checked. For the contesting concerning $(n_i, \widetilde{n}_i)$, the preceding counter must be considered to compute the correct value of the contested bit, which allows to decide if the contesting is valid or not. The computation of that bit will of course depend on the nature of the contesting.

For a contesting ? on a tuple of letters, the corresponding tuple in the preceding configuration has to be considered. To find it, one uses the values of the counters that must therefore be correct. For this reason, the validity of the counters has to be checked first and only after this, can one check the validity of the tuples. Therefore, in the winning condition $\Omega^{ext}_{\mathcal{A}_1 \rhd \mathcal{A}_2}$, $\mathcal{A}_1$ will be used to check the validity of the counters and $\mathcal{A}_2$ to check the validity of the configurations.

The automaton $\mathcal{A}_1$ copies its input word (in our case the description of the run of M) in its stack. If it eventually reads a contesting $!_v$ or $\widetilde{!_v}$, it determines the index j of the contested bit (which is bounded by N) by popping and counting the bits already written in its stack until it finds a tuple of letters. Figure 5 illustrates the next two cases. If the contesting is $!_v$, $\mathcal{A}_1$ pops the next j pairs of bits of its stack and considers their second components to compute the j-th bit of the counter obtained by adding 1 to the top counter of the stack, and thus decides whether or not there was an error. If the contesting was $\widetilde{!_v}$, it pops and ignores the j first pairs and then verifies with the first components

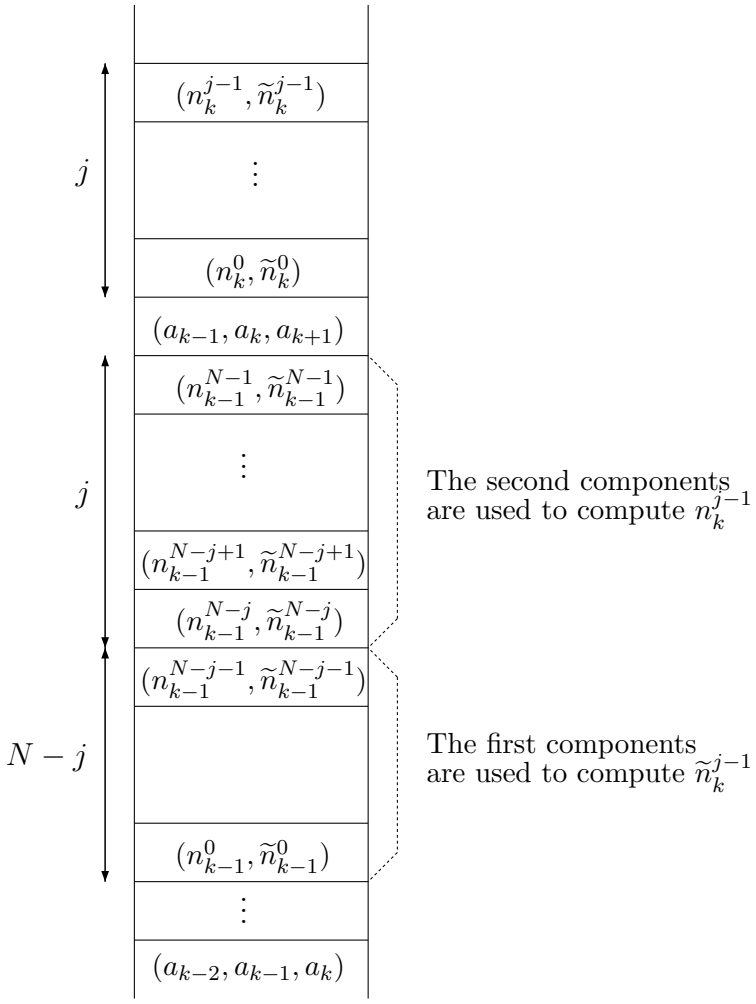


Fig. 5. Stack content of $\mathcal{A}_1$ when reading $!_v$ or $\widetilde{!}_v$

of the next $N - j$ bits stored in the stack whether or not there was an error. After having checked the correctness of the contesting, $\mathcal{A}_1$ pushes an infinite sequence of ♠ if the contesting was correct, and pushes an infinite sequence of ♡ otherwise.

If $\mathcal{A}_1$ reads a contesting $!_c$ or $\widetilde{!}_c$, it pops the last N bits in its stack and compares the new top bit with the contested one. The contesting is correct if and only if they are different. This case is illustrated by Figure 6. After having checked the correctness of the contesting, $\mathcal{A}_1$ pushes an infinite sequence of ♠ if the contesting was correct, and pushes an infinite sequence of ♡ otherwise.

The automaton $\mathcal{A}_2$ copies its input word in its stack. It goes in a final state when it reads ♡ and goes and stays forever in a non final state when it reads ♠. In particular, it accepts any word ending by an infinite sequence of ♡ and rejects any word ending by an infinite sequence of ♠. Note that the limits of the stack contents of $\mathcal{A}_1$ can only contain a contesting of the form ?. Figure 7 illustrates the behavior of $\mathcal{A}_2$. When $\mathcal{A}_2$ reads some contesting ?, it pops until

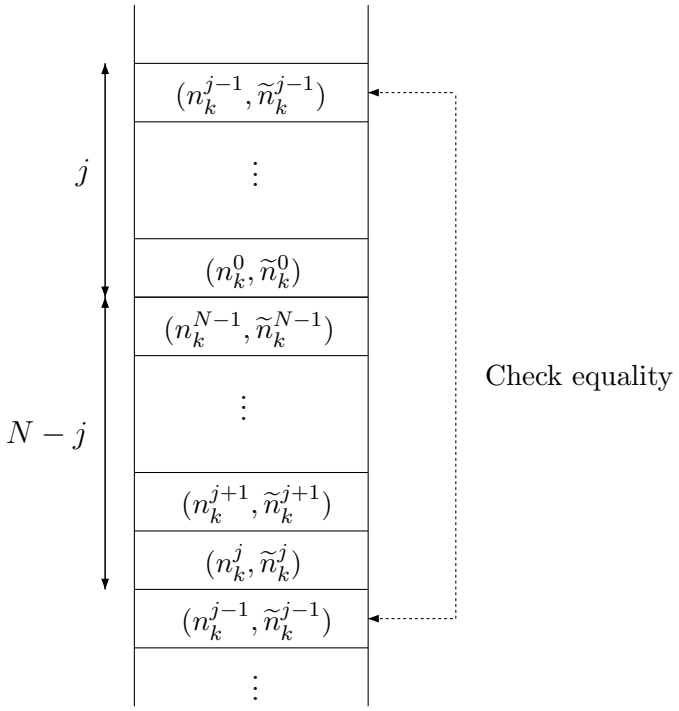


Fig. 6. Stack content of $\mathcal{A}_1$ when reading $!_c$ or $\widetilde{!}_c$

it finds a symbol describing a transition of the Turing machine and remembers it. Then it reads synchronously the first components of the counters described after ? and pops its stack and focuses on the second component of the counters. It does so until it finds a correspondence between the counters. Note that for this search, it only compares the counter it reads with the first copy of the counter it has in its stack (otherwise it might run out of copies before finding the corresponding counter). This search terminates because there are 2^N possible values for a counter and the counter in the input word is repeated 2^N times. Therefore, there are enough copies to successfully perform an exhaustive search. Once the corresponding counter is found, the top stack symbol is the tuple of letters whose index is the same as the one being contested. The automaton $\mathcal{A}_2$ can therefore decide whether its central letter has been correctly updated and thus concludes on the validity of the contesting. Then it stops reading its input word and goes in an accepting state if the contesting was not correct, and loops in a non accepting state otherwise. Finally, in the case where no correspondence was found, it loops in an non accepting state. In that case, Eve did an error in the description of the configurations of the Turing machine.

One can prove that this reduction is of polynomial size, and is such that Eve has a winning strategy if and only if the Turing machine accepts from the empty tape. The winning strategy for Eve is to describe an accepting run of the Turing machine. If the Turing machine rejects from the empty tape, a winning strategy for Adam is to describe a rejecting run and to contest

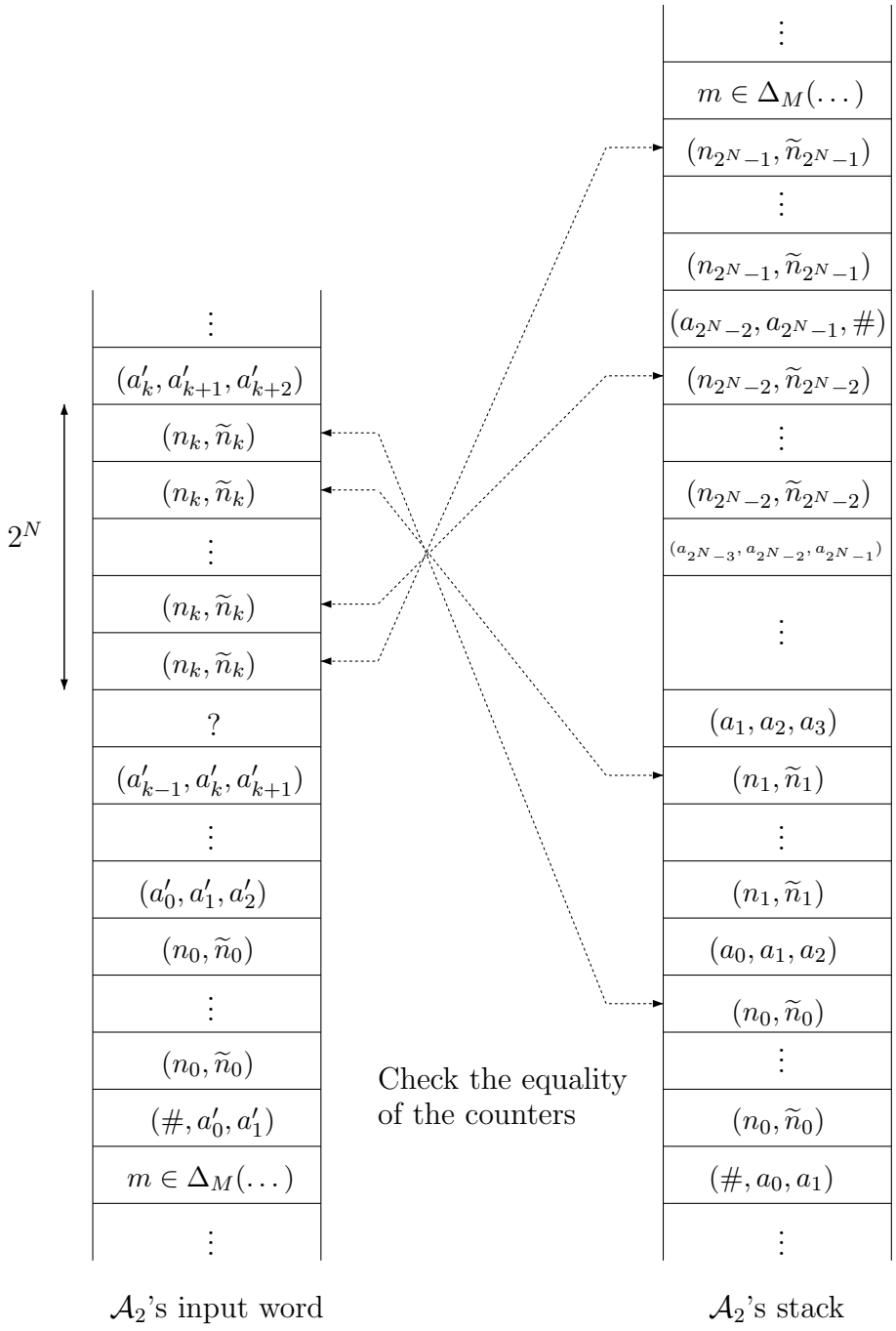


Fig. 7. Behavior of $\mathcal{A}_2$ after reading ?

whenever Eve cheats in the description of the run.

5.2.4 General Case: $k \geq 1$

In this section, we explain how to extend the techniques used in section 5.2.3 to prove the $(k+1)$-EXPTIME hardness of the winning conditions of the form $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$. For this, we simulate an alternating Turing machine M

using $tow(k, N)$ space by a game on a finite game graph equipped with such a winning condition.

As in the case $k = 1$, Eve and Adam will describe a branch of a run of the Turing machine by writing sequences of configurations. Previously, a configuration was represented by a sequence of the form

$$(\#a_0a_1)(n_0, \widetilde{n}_0)^{2^N}(a_0a_1a_2)(n_1, \widetilde{n}_1)^{2^N} \dots (a_{2^N-2}a_{2^N-1}\#)(n_{2^N-1}, \widetilde{n}_{2^N-1})^{2^N}.$$

Now, the length of a configuration is no longer exponential but k-times exponential. A first idea would be to keep the same representation, except that the counters $(n_i, \widetilde{n}_i)$ would be bigger. Unfortunately, their length would no longer be linear (but $(k-1)$-times exponential), and therefore it would no longer be possible, with polynomial size memory, to check whether they represent the correct integers.

To address this problem, we adopt a more refined representation.

Definition 41 (h-exp decomposition of an integer) *The 1-exp decomposition of an integer $R < tow(1, N)$ is the sequence $((b_{N-1}, b_0) \dots (b_1, b_{N-2})(b_0, b_{N-1}))^{tow(1,N)}$, where $b_{N-1} \dots b_1b_0$ is the binary representation of R with the most significant bit on the left and therefore $b_0b_1 \dots b_{N-2}b_{N-1}$ is the binary representation of R with the most significant bit on the right.*

Let $h > 1$ and let R be some integer such that $R < tow(h, N)$. The h-exp decomposition of R is the sequence:

$$\Big[(b_0, b_{tow(h-1,N)-1})\alpha_0(b_1, b_{tow(h-1,N)-2})\alpha_1 \cdots (b_{tow(h-1,N)-1}, b_0)\alpha_{tow(h-1,N)-1}\Big]^{tow(h,N)}$$

where

- *$b_{tow(h-1,N)-1} \dots b_1b_0$ is the binary representation of R with the most significant bit on the left;*
- *$b_0 \dots b_{tow(h-1,N)-2}b_{tow(h-1,N)-1}$ is the binary representation of R with the most significant bit on the right;*
- *α_i is the $(h-1)$-exp decomposition of i.*

For convenience, we assume that the binary alphabets used in the h-exp decomposition are distinct for any level h so that the decomposition is easily readable.

Finally, we designate the sequence $(b_0, b_{tow(h-1,N)-1}) \dots (b_{tow(h-1,N)-1}, b_0)$ as a counter of level h.

A configuration $a_0a_1 \cdots a_{tow(k,N)-1}$ of the Turing machine M is then represented by the following word

$$(\#a_0a_1)n_0(a_0a_1a_2)n_1 \ldots (a_{(tow(k,N)-2)}a_{(tow(k,N)-1)}\#)n_{tow(k,N)-1},$$

where n_i is the k-exp decomposition of i. Note that for $k = 1$, we obtain exactly the representation used in the preceding section.

Let us now explain how a play goes on. Eve describes the initial configuration. Then, depending on whether the control state in this configuration is existential or universal, Eve (existential state) or Adam (universal state), chooses a move, that is, goes trough an edge whose label is a description of a transition of the Turing machine. Then, Eve describes the resulting configuration and so on. If an accepting configuration is eventually reached, then the play goes in a special vertex where it loops forever trough an edge labeled by a special symbol $\heartsuit$.

To prevent Eve from cheating, there are special symbols $!_c^h, !\widetilde{_c^h}, !_v^h, !\widetilde{_v^h}$ for all $h = 1 \ldots k$ (c stands for copy and v for value) and ? that Adam can write whenever he wants to contest the validity of the last symbol written by Eve. They can be used in the following cases:

- The j-th pair of bits of a counter of level h is false. If it is the first copy of the counter, the value should be equal to the value of the preceding counter plus 1. Therefore, Adam writes $!_v^h$ (if the mistake is for the representation with the most significant bit on the left) or $!\widetilde{_v^h}$ (otherwise) just after the wrong pair of bits. If it is ont the first copy, the error concerns the duplication. In that case, Adam writes $!_c^h$ (if the mistake is for the representation with the most significant bit on the left) or $!\widetilde{_c^h}$ (otherwise) just after the wrong pair of bits.
- The value of some tuple $(a_{i-1}a_ia_{i+1})$ is incorrect. In that case, Adam writes ? just after it.

Once a contesting on a counter of level 1 has been made, the play goes in a special vertex where it loops forever trough an edge labeled by a special symbol $\spadesuit$. In the case of a contesting concerning a counter of level $h \geq 2$, the $(h-1)$-exp decomposition of the index of the contested bit is written (and can be contested) and then the symbol $\spadesuit$ is written forever. In the case of a contesting ?, the value of the k-decomposition of the index of the contested tuple is written (and can be contested) and then the symbol $\spadesuit$ is written forever.

The alternation between both players (that allows Adam to contest any symbol written by Eve), the memory to decide whether the control state is existential

or universal, and the memory of a contesting or a final configuration (to write ♠ or ♡) are encoded in the vertices. It is also the same to force Eve to start by describing the k-exp decomposition of 0, and to describe an overlapping sequence of tuples of letters. More precisely, if the last tuple was (a_{i-1}, a_i, a_{i+1}), then Eve is forced to write as next tuple (a_i, a_{i+1}, a_{i+2}) for some letter a_{i+2}.

Let us now describe how the validity of a contesting is checked. For the contesting concerning a counter, the preceding counter must be considered to compute the correct value of the contested bit, which allows to conclude. The computation of that bit will of course depend on the nature of the contesting. For a contesting ? on a tuple of letters, the corresponding tuple in the preceding configuration has to be considered. To find it, one uses the values of the counters that must therefore be correct. For this reason, the validity of the counters of level 0 is checked first, then the validity of counters of level 1 is checked and so on. Finally, the validity of the configurations is checked. Therefore, in the winning condition $\Omega^{ext}_{\mathcal{A}_1 \triangleright \ldots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$, $\mathcal{A}_h$ will be used to check the validity of the counters of level $h \leq k$ and $\mathcal{A}_{k+1}$ to check the validity of the configurations.

Now, let us describe how the different pushdown automata behave.

- The automaton $\mathcal{A}_1$ copies its input word in its stack. If it reads a contesting $!^1_c$ or $\widetilde{!^1_c}$, it pops the last N bits in its stack and compares the new top bit with the contested one. The contesting is correct if and only if they are different. If the contesting was correct, it stops reading the word and pushes an infinite sequence of ♠, otherwise it stops reading and pushes an infinite sequence of ♡.

  If it eventually reads a contesting $!^1_v$ or $\widetilde{!^1_v}$, it determines the index j of the contested bit (which is bounded by N) by popping and counting the bits already written in its stack until it finds a bit from a counter of level 2. If the contesting was $!^1_v$, it pops the next j first pairs of bits and considers their second components to compute the j-th bit of the counter obtained by adding 1 to the top counter (of level 1) of the stack. Then it can decide whether or not there was an error. If the contesting was $\widetilde{!^1_v}$, it pops and ignores the j first pairs and then verifies, with the first components of the next $N - j$ bits stored in the stack whether there was or not an error. After having checked the correctness of the contesting, $\mathcal{A}_1$ pushes an infinite sequence of ♠ if the contesting was correct, and pushes an infinite sequence of ♡ otherwise.
- The automaton $\mathcal{A}_2$ copies its input word in its stack. If it reads a contesting $!^2_c$ or $\widetilde{!^2_c}$, it has to pop its stack until it finds the bit whose index in the preceding counter of level 2 is the same as the one being contested. In order to do this, it reads the next 1-decomposition (which is a repeated binary representation of the index of the contested bit) and at the same time, it pops its stack and compares what it reads with the various indices it has

in its stack (note that in the stack the indices are also represented in a iterated form: $\mathcal{A}_2$ only compares to the first representation). In order to do this, it compares the first component (most significant bit first) of what it reads with the second component (less significant bit first) of what it pops. Eventually, it finds the corresponding bit of the preceding counter and compares it with the contested one. The contesting is correct if and only if they are not equal. If the contesting was correct, it stops reading the word and pushes an infinite sequence of ♠, otherwise it stops reading and pushes an infinite sequence of ♡.

If it finds a contesting $!_v^2$ or $!_v^{\widetilde{2}}$, let j be the index of the contested pair of bits (which is bounded by $(tow(1,N)-1)$). To check the validity of the contesting, one has to compute the bit from the preceding value of the counter. The computation depends on whether the contested bit is the first or the second of the pair. If the contesting is $!_v^2$, it is sufficient to consider the second component of the pairs of indices $(tow(1,N)-1)\dots(tow(1,N)-1-j)$ of the preceding counter. If the contesting is $!_v^{\widetilde{2}}$, it is sufficient to consider the first component of the pairs of indices $(tow(1,N)-1-j)\dots 0$ of the preceding counter. Note that the binary representation of $tow(1,N)-1-j$ is obtained from the binary representation of j by changing all 0 by 1 and all 1 by 0. Therefore, the binary representation of $(tow(1,N)-1-j)$ can be computed on the fly from the one of j (which is read by $\mathcal{A}_2$). Both cases are thus almost the same as for the preceding contestings, except that an on the fly addition is performed while searching. Thus, these cases are no more difficult than the preceding ones. If the contesting was correct, $\mathcal{A}_2$ stops reading the word and pushes an infinite sequence of ♠, otherwise it stops reading and pushes an infinite sequence of ♡.

If, at any moment, $\mathcal{A}_2$ reads some symbol ♠ (there was a valid contesting on a counter of level 1), it stops reading and pushes an infinite sequence of ♠. Symmetrically, if $\mathcal{A}_2$ reads some symbol ♡, it stops reading and pushes an infinite sequence od ♡.

- Let $3 \leq h \leq k$. $\mathcal{A}_h$ works as $\mathcal{A}_2$ except that it handles on the contesting $!_c^h$, $!_c^{\widetilde{h}}$, $!_v^h$, $!_v^{\widetilde{h}}$. Another difference is that $\mathcal{A}_h$ copies the word it reads in its stack, but it skips all bits concerning counters of level $h-2$ as they are no longer useful. Therefore, in its stack, there are only informations on counters of level greater or equal than $h-1$.
- $\mathcal{A}_{k+1}$ copies the word, it reads in its stack, but it skips all bits concerning counters of level $k-1$. It goes in a final state when it reads ♡ and goes and stays forever in a non final state when it reads ♠. In particular, it accepts words ending by an infinite sequence of ♡ and rejects words ending by an infinite sequence of ♠. Note that the limits of stack contents that $\mathcal{A}_{k+1}$ reads can only contain a contesting of the form ?. When $\mathcal{A}_{k+1}$ reads some contesting ?, it reads the next symbol (which is a tuple of bits) and pops in its stack until it finds a symbol describing the transition of the Turing machine and remembers it. Then, it reads synchronously the first component

of the counters described after ?, and pops in the stack and considers on the second component of the counters. It does so until it finds a correspondence between the counters. Again, this exhaustive search works since there are enough copies of the searched counter in the input word (when popping $\mathcal{A}_{k+1}$ only compares to the first value of the iterated counter it reads in the stack). Then, as top symbol, it gets the tuple of letters whose index is the same as the one being contested. It can therefore check whether there was an error for the central letter of the tuple. In case of error, it stops reading and loops in a non accepting state, otherwise, it stops reading and goes in an accepting state. In the case where no correspondence was found, it loops in a non accepting state.

One can prove that the preceding reduction is of polynomial size and is such that Eve has a winning strategy if and only if the Turing machine accepts from the empty tape. The winning strategy for Eve is to describe an accepting run of the Turing machine. If the Turing machine rejects from the empty tape, a winning strategy for Adam is to describe a rejecting run and to contest if Eve cheats in the description of the run.

6 Winning Positions and Strategies

6.1 Winning Positions

In this section, we give results on the set of winning positions for pushdown games equipped with winning conditions of the form $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$.

In [18,3], it is shown that the set of winning positions in a parity pushdown game is a regular language. In fact, using the same techniques, one can prove similar results for various winning conditions [19]. For instance: unboundedness, strict unboundedness, trampoline (Eve wins if and only if some configuration is infinitely visited while the stack is unbounded) or for more exotic winning conditions (for instance: if n_i denotes the number of distinct palindromes appearing in the stack of the i-th configuration of a play, Eve wins if and only if $(n_i)_{i \geq 1}$ is unbounded).

The main idea to prove these results is to note that the winner does not depend on a prefix of the play neither on a finite number of letters on the bottom of the stack, provided these letters will never appear on top of the stack. This second point is no longer true for the winning conditions $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$. Indeed, if some extra letter is added somewhere in the stack, the stack limit is modified, even if this letter never appears on top of the stack. Therefore, the winner of the play may changed.

In fact, for the winning conditions of the form $\Omega^{int}_{\mathcal{A}}$, the set of winning positions may not be regular. For instance, every deterministic context-free language may occur as a winning set.

Proposition 42 *Let $\mathcal{A}$ be some deterministic pushdown automaton on finite words. There exists a deterministic Büchi automaton $\mathcal{B}$, a pushdown process $\mathcal{P} = \langle Q, \Gamma, \bot, \Delta\rangle$, a state $q \in Q$ and a partition $Q = Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ such that, in the induced pushdown game equipped with the winning condition $\Omega^{int}_{\mathcal{B}}$, the set $\{u \mid (q,u) \in W_{\mathbf{E}}\}$ is exactly the language accepted by $\mathcal{A}$.*

PROOF. Let A be the alphabet of $\mathcal{A}$ and let $\# \notin A$ be a new symbol. We set $\mathcal{P} = \langle \{p,q\}, A \cup \{\#\}, \bot, \Delta\rangle$ where Δ is defined by: $\Delta(q,a) = \{push(p,\#)\}$ for all letter $a \in A$ and $\Delta(p,\#) = \{push(p,\#)\}$. In other words, $\mathcal{P}$ is deterministic, and pushes an infinite sequence of # on top of its initial stack contents. $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ is any partition of Q. $\mathcal{B}$ works as $\mathcal{A}$, except that it loops when it reads # and cannot read any letter after having read #. Therefore, $\mathcal{B}$ accepts the words of the form $u\#^{\omega}$, where u is accepted by $\mathcal{A}$. □

It remains open whether there exists a pushdown game equipped with a winning condition of the form $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ such that the set of winning positions for Eve is not a deterministic context-free language.

6.2 Strategies

In this section, we discuss the nature of the winning strategies in the pushdown games equipped with winning conditions of the form $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. In particular, we show that there are persistent strategies in these games.

First, let us recall the definition of a persistent strategy [14].

Definition 43 (Persistent strategy) *A strategy φ for Eve is persistent if for any partial play $v_1 v_2 \cdots v_k$ where Eve respects φ, if $v_i = v_j$, for some $1 \leq i$, $j < k$, and if v_i is a vertex where Eve is to move, then $v_{i+1} = v_{j+1}$.*

In other words, a persistent strategy may require memory, but once a choice is made, it is forever. In this section, we show that for the pushdown games equipped with winning conditions of the form $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ Eve has a persistent winning strategy.

Theorem 44 *Let $\mathcal{G}$ be a pushdown game graph and let $\mathbb{G} = (\mathcal{G}, \Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}})$ be a game on $\mathcal{G}$ equipped with some winning condition $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. Then,*

Eve has a persistent winning strategy from any winning position for her in the pushdown game $\mathbb{G}$.

Before proving Theorem 44, let us give some definitions.

Definition 45 (Subword) *Let $u = a_0a_1a_2a_3\cdots$ be an infinite word and let $(i_j)_{j\geq 0}$ be a strictly increasing sequence of integers. The word $v = a_{i_0}a_{i_1}a_{i_2}a_{i_3}\cdots$ is called a* subword *of u.*

Definition 46 (Strategy Tree) *Let φ be a strategy for Eve in a game $\mathbb{G} = (\mathcal{G}, \Omega)$ played on an unlabeled game graph $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ and equipped with an internal winning condition Ω. Let $V = V_{\mathbf{E}} \cup V_{\mathbf{A}}$ and let $v \in V$. One associates with φ and v an infinite tree T_φ with nodes labeled by V in the following way. The root of T_φ is labeled by v. For any node in T_φ, labeled by some element $w \in V$ one has:*

- *If $w \in V_{\mathbf{E}}$, the node has a unique successor labeled by $\varphi(\lambda)$ where λ is the labeling of the unique path from the root of T_φ to the current node.*
- *If $w \in V_{\mathbf{A}}$, let $\{w_1, w_2, \ldots, w_k\}$ be the set of successors of w in G. Then the node has k successors labeled by $w_1, w_2, \ldots w_k$.*

Therefore any labeling of an infinite path in T_φ is a play in $\mathcal{G}$ starting from v where Eve respects φ. Conversely, any such play is the labeling of some infinite path in T_φ.

The following result directly implies Theorem 44.

Proposition 47 *Let $\mathbb{G} = (\mathcal{G}, \Omega)$ be a game on an unlabeled game graph $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ equipped with an internal winning condition Ω. Assume that Ω satisfies the two following conditions:*

(1) For all winning play $\lambda \in \Omega$, and for all vertex $v \in V = V_{\mathbf{E}} \cup V_{\mathbf{A}}$, v appears only finitely often in λ.
(2) For all winning play $\lambda \in \Omega$, any infinite subword λ' of λ is in Ω.

Then, from any winning position for Eve, she has a winning strategy which insures that every vertex is visited at most once in a play. In particular, this strategy is a persistent one.

PROOF. Let v be a winning position for Eve and let φ be a winning strategy for her from v. φ can be represented as an infinite V-labeled tree T_φ describing all the plays where Eve follows φ. Condition (1) implies that for every label $w \in V$ appearing in T_φ, there is some infinite subtree of T_φ, which root is labeled by w and in which w only labels the root. Indeed, assume by contradiction, that any infinite subtree of T_φ which root is labeled by w contains another node

labeled by w. Let T_1 be some infinite subtree of T_φ which root is labeled by w. T_1 contains therefore another node labeled by w. Let T_2 be the infinite subtree of T_1 (and therefore of T) rooted in that node. Iterating the construction, one obtains an infinite sequence $T_\varphi \supsetneq T_1 \supsetneq T_2 \supsetneq \cdots$ of infinite subtrees ,which roots are all labeled by w. The infinite path from the root of T_φ that visits the roots of the trees $(T_i)_{i\geq 1}$ is an infinite path in T_φ that contains infinitely many nodes labeled by w. As the labeling of such a path is a winning play in $\mathbb{G}$, it leads a contradiction with (1).

Now, let us describe how to construct from φ a winning strategy φ' for Eve that does not visit a vertex twice. The strategy φ' maintains an infinite tree T as memory. At the beginning, $T = T_\varphi$. From a position $w \in V_{\mathbf{E}}$ and a tree T, φ' considers a subtree of T which root is labeled by w, and where w only labels the root. This tree becomes the new memory T, and the move given by φ' is to the vertex labeling the (unique) son of the root of T. One easily shows that φ' is always defined, and insures that no vertex is visited twice. In addition, any play, where Eve respects φ' is a subword of a play where Eve respects φ, and is therefore winning by condition (2). □

6.3 Effective Strategies

In this section, we informally discuss the following questions: do our methods provide effective winning strategies? Which kind of machine model may be used to give a finite description of a winning strategy?

In the proof of Theorems 26 and 27, we have shown how to construct an effective strategy in the game played on a finite graph from an effective strategy in the equivalent pushdown game. The main idea was to build, from the current partial play, an equivalent partial play in the pushdown game. This was done by a function τ that computes the run of some deterministic pushdown automaton on the labeling of the current play. Then, the value of the strategy in the pushdown game on this partial play was considered. Now, assume that the strategy in the pushdown game only needs a stack, therefore, the strategy we obtain by this construction for the game on the finite graph only needs a stack (which alphabet is augmented by a finite number of symbols used to simulate τ). For instance, as pushdown parity games admits effective stack strategies [24], it implies that there are effective stack strategies in a the game played on a finite graph equipped with a condition of the form $\Omega^{ext}_{\mathcal{A}_1}$.

Now, consider the reduction from the pushdown game to a game played on a finite graph. In the converse implication of the proof of Theorem 28, we construct a strategy in the pushdown game from a strategy in the game played on the finite graph. For this, a stack Υ is used to store a play in the finite

graph. Then, the value of the strategy in the finite graph on the partial play stored in the stack is used to decide how to play. Now, assume that the strategy in the finite graph uses a stack on some alphabet S. Then, one can encode and update this into the stack Υ which alphabet is now augmented by stacks on the alphabet S: Υ is a stack (on an alphabet S) containing stacks and other symbols.

Iterating this reasoning shows that the preceding constructions provide effective winning strategies. Moreover, an high-order stack (that is a stack which elements are also stacks) is sufficient as a memory for these strategies. The number of nested stacks is $n+2$ if the winning condition that we consider is of the form $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ or $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_{n+1} \triangleright \mathcal{A}_{n+2}}$.

7 Conclusion and Perspectives

We have provided a family of winning conditions that have an arbitrary finite Borel complexity while remaining decidable for pushdown games and games on finite graphs. The complexity of deciding the winner for such a winning condition is a non-elementary problem that is elementary-hard. In addition, for pushdown games, it gives an example of decidable winning conditions inducing non regular sets of winning positions. The exact form of the winning sets remains open. Finally, we have shown that there are persistent winning strategies for pushdown games equipped with these winning conditions. The existence of positional strategies remains open.

Studying the classes $(\mathbb{C}_n(A))_{n\in\mathbb{N}}$ is also an interesting question. As they contain languages of arbitrary finite Borel complexity, these classes are not included in the one of determinisitic ω-context-free languages (which only contains languages in $\mathcal{B}(\Sigma_2)$). Are they a strict subclass of ω-context-free languages? Note that in case of inclusion it would be strict as there are ω-context-free languages that are not Borel sets. From the game point of view, studying the closure properties of these classes under boolean operations is also relevant. Finally, let us mention that the decidability of the emptiness problem and the universality problem for languages of these classes directly follows from the decidability of the games on finite graphs.

Acknowledgments

I gratefully acknowledge Jacques Duparc for suggesting me to study this family of conditions. His advice and knowledge of Borel complexity were very important in this research. I would also like to express my thanks to Anca

Muscholl and to Solveig, for their help while writing this paper, and to the anonymous referees for their remarks.

References

[1] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controlers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

[2] A. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with the unboundedness and regular conditions. In *Proceedings of FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference*, volume 2914 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2003.

[3] T. Cachat. Uniform solution of parity games on prefix-recognizable graphs. In *Proceedings of the 4th International Workshop on Verification of Infinite-State Systems*, volume 68 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.

[4] T. Cachat, J. Duparc, and W. Thomas. Solving pushdown games with a Σ_3-winning condition. In *Proceedings of the 11th Annual Conference of the European Association for Computer Science Logic, CSL 2002*, volume 2471 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2002.

[5] R.S. Cohen and A.Y. Gold. ω-computations on deterministic pushdown machines. *Journal of Computer and System Sciences*, 3:257–300, 1978.

[6] J. Duparc. Wadge hierarchy and Veblen hierarchy. part I: Borel sets of finite rank. *Journal of Symbolic Logic*, 66(1):56–86, 2001.

[7] E.A. Emerson, C.S. Jutla, and A.P. Sistla. On model-checking for the mu-calculus and its fragments. *Theoretical Computer Science*, 258(1-2):491–522, 2001.

[8] O. Finkel. Topological properties of omega context-free languages. *Theoretical Computer Science*, 262:669–697, 2001.

[9] H. Gimbert. Parity and exploration games on infinite graphs. In *Proceedings of the 13th Annual Conference of the European Association for Computer Science Logic, CSL 2004*, volume 3210 of *Lecture Notes in Computer Science*, pages 56–70. Springer, 2004.

[10] M. Jurdziński. Small progress measures for solving parity games. In *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301, Lille, France, February 2000. Springer-Verlag.

[11] A.S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate texts in mathematics*. Springer Verlag, 1994.

[12] O. Kupferman and M.Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Proceedings of CAV'00*, volume 1855 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2000.

[13] C. Löding. Private communication.

[14] J. Marcinkowski and T. Truderung. Optimal complexity bounds for positive LTL games. In *Proceedings of the 11th Annual Conference of the European Association for Computer Science Logic, CSL 2002*, volume 2471 of *Lecture Notes in Computer Science*, pages 262–275. Springer, 2002.

[15] D.A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[16] Christos H. Papadimitriou. *Complexity Theory*. Addison Wesley, 1994.

[17] D. Perrin and J-E. Pin. *Infinite words*, volume 141 of *Pure and applied mathematics*. Elsevier, Academic Press, 2004.

[18] O. Serre. Note on winning positions on pushdown games with ω-regular conditions. *Information Processing Letters*, 85:285–291, 2003.

[19] O. Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris 7, 2004.

[20] W. Thomas. On the synthesis of strategies in infinite games. In *Proceedings of STACS '95*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995.

[21] W. Thomas. Infinite games and verification (extended abstract of a tutorial). In *Proceedings of the International Conference on Computer Aided Verification CAV'02*, volume 2404 of *Lecture Notes in Computer Science*, pages 58–64. Springer, 2002.

[22] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games (Extended abstract). In E. A. Emerson and A. P. Sistla, editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215, Chicago, IL, USA, July 2000. Springer-Verlag.

[23] W. W. Wadge. *Reducibility and determinateness of the Baire space*. PhD thesis, Berkeley, 1984.

[24] I. Walukiewicz. Pushdown processes: games and model checking. *Information and Computation*, 157:234–263, 2000.

[25] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.