Winning regions of higher-order pushdown games

Arnaud Carayol, Matthew Hague, Antoine Meyer, Luke Ong, Olivier Serre

Winning regions of higher-order pushdown games*

A. Carayol† M. Hague‡ A. Meyer§ C.-H. L. Ong¶ O. Serre‖

Abstract

In this paper we consider parity games defined by higher-order pushdown automata. These automata generalise pushdown automata by the use of higher-order stacks, which are nested "stack of stacks" structures. Representing higher-order stacks as well-bracketed words in the usual way, we show that the winning regions of these games are regular sets of words. Moreover a finite automaton recognising this region can be effectively computed.

A novelty of our work are abstract pushdown processes which can be seen as (ordinary) pushdown automata but with an infinite stack alphabet. We use the device to give a uniform presentation of our results.

From our main result on winning regions of parity games we derive a solution to the Modal Mu-Calculus Global Model-Checking Problem for higher-order pushdown graphs as well as for ranked trees generated by higher-order safe recursion schemes.

1 Introduction

Higher-order pushdown automata were introduced by Maslov [31] as a generalisation of pushdown automata and nested pushdown automata to extend the indexed languages introduced by Aho [1]. Whereas an ordinary (*i.e.* order-1) pushdown automaton works with a stack of symbols (*i.e.* order-1 stack), a pushdown automaton of order 2 works with a stack of (order-1) stacks. In addition to pushing a symbol onto and popping a symbol from the top-most order-1 stack, an order-2 pushdown automaton can duplicate or remove the entire top-most (order-1) stack. Pushdown automata of higher orders are defined in a similar way and have been extensively studied as language acceptors [15, 17].

Recently, the infinite structures defined by these automata have received a lot of attention. In [28], the families of infinite terms defined by higher-order pushdown automata were shown to correspond to the solutions of safe higher-order recursion schemes. Subsequently, in [14, 13], the ε-closure of their configuration graphs were shown to be exactly those constructible from finite graphs using natural graph transformations (see [36] for a survey).

A remarkable property of these graphs is that we can decide the validity of any formula of monadic second-order (MSO) logic. Unfortunately the decision procedure is non-elementary (in the size of the formula) and this is already so in the case of pushdown graphs. In order to obtain an elementary decision procedure, we consider the μ-calculus: a weaker modal logic equi-expressive with MSO over trees [26]. The two main algorithmic problems in this setting are the *(local) model-checking problem* (*i.e.* to decide if a particular configuration satisfies a given μ-calculus formula) and the *global model-checking problem* (*i.e.* to compute a finite description of the set of configurations satisfying a given formula). To solve these problems, we consider the associated two-player parity game, whose size is polynomial in both the size of the automaton and of the formula [16]. The two versions of the model-checking problem above are respectively equivalent to deciding which player wins from a given configuration and to giving a finite description of the winning region for each player.

For parity games defined by pushdown automata, Walukiewicz has given an ExpTime decision procedure to compute the winner from a given configuration [38]. In [6, 32], the winning region is shown to be regular when a configuration (q, w) is represented by the word qw. Note that this result can easily be derived from the results by Vardi [37]. For order-n pushdown automata, an n-ExpTime decision procedure for the local version of the problem was given by Cachat [7] using techniques from [37]. In this article, we shall consider the global version of the problem *i.e.* the computation of a finite representation of the winning region, and we obtain, as a by-product, a new proof of Cachat's result.

To give a finite description of a set of higher-order stacks,

*We direct readers to the (downloadable) long version [11] of this paper in which all proofs are presented.

†Arnaud.Carayol@univ-mlv.fr IGM (Université Paris Est & CNRS)

‡Matthew.Hague@comlab.ox.ac.uk Oxford University Computing Laboratory (OUCL)

§Antoine.Meyer@liafa.jussieu.fr LIAFA (Université Paris Diderot – Paris 7 & CNRS)

¶Luke.Ong@comlab.ox.ac.uk OUCL

‖Olivier.Serre@liafa.jussieu.fr LIAFA

we represent these stacks as well-bracketed words. Note that the depth of the bracketing is bounded by the order of the stack. By extension, we say that a set of higher-order stacks is regular if the set of associated well-bracketed words is regular. Our main result is that the winning regions of parity games over higher-order pushdown graphs are regular. Moreover we can construct in n-EXPTIME a finite deterministic automaton accepting it.

To simplify the presentation, we consider a more general notion of pushdown automata, called *abstract pushdown automata*, which work with a possibly infinite stack alphabet. Standard pushdown automata are abstract pushdown automata with a finite stack alphabet; order-$(k+1)$ pushdown automata are abstract pushdown automata whose stack alphabet is the (infinite) set of order-k stacks. Our main technical result concerns parity games over the configuration graphs of abstract pushdown automata. From an abstract pushdown parity game, we construct a reduced parity game based on the stack alphabet (*i.e.* which does not make use of a stack structure) and we show that a finite description of the winning region of the original game can be derived from a finite description of the winning region of the reduced game. Applied to higher-order pushdown automata, this result allows us to reduce the problem of computing the winning region of an order-$(k+1)$ pushdown parity game to that of computing the winning region of an order-k pushdown parity game. We also show that starting from a winning strategy in the reduced parity game one can effectively build an abstract pushdown automaton that realises a winning strategy in the original game and whose stack is synchronised with the one used in the game. Applied to higher-order pushdown parity games, this result allows us to prove that any such game always admits an effective winning strategy realised by a higher-order pushdown automata of the same order and whose stack is synchronised with the one used in the game.

As an application of these results, we solve the μ-calculus global model-checking problem for higher-order pushdown graphs, and for ranked trees generated by safe higher-order recursion schemes.

Related work. In [4], Bouajjani and Meyer considered simpler reachability games over higher-order pushdown automata with a single control state. They showed that the winning regions in these games are regular. In [25], Hague and Ong extended the result to arbitrary higher-order pushdown automata. The results presented here have been obtained by Hague and Ong, and also (separately and independently) by the other co-authors. The present article follows the latter approach whereas the proof in [23] generalises the saturation method developed in [25].

A similar result was obtained in [12] for a stronger notion of regularity (introduced in [10, 20]) which coincides with MSO-definable sets of configurations. In contrast, the simpler notion presented here can only capture properties definable in μ-calculus. In particular, owing to the higher-order $push$ operations, the set of configurations reachable from a given configuration is not regular in the sense of the present paper. (To our knowledge the results presented here do not appear to be derivable from Carayol's work [10].)

2 Definitions

An *alphabet* A is a (possibly infinite) set of letters. In the sequel A^* denotes the set of *finite words* over A and A^ω the set of *infinite words* over A. The empty word is denoted by ε.

Infinite two-player games. Let $G = (V, E)$ be a (possibly infinite) graph with vertex-set V and edge-set $E \subseteq V \times V$. Let $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ be a partition of V between two players, Éloïse and Abelard. A *game graph* is a tuple $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$. An *infinite two-player game* on a game graph $\mathcal{G}$ is a pair $\mathbb{G} = (\mathcal{G}, \Omega)$, where $\Omega \subseteq V^\omega$ is a *winning condition*.

Éloïse and Abelard play in $\mathbb{G}$ by moving a token between vertices. A *play* from some initial vertex v_0 proceeds as follows: the player owning v_0 moves the token to a vertex v_1 such that $(v_0, v_1) \in E$. Then the player owning v_1 chooses a successor v_2 and so on. If at some point one of the players cannot move, she/he loses the play. Otherwise, the play is an infinite word $\Lambda \in V^\omega$ and is won by Éloïse if and only if $\Lambda \in \Omega$. Nevertheless, for all game graphs considered in this article one can always assume without loss of generality that they have no dead-ends. A *partial play* is any prefix of a play.

A strategy for Éloïse is a function assigning, to any partial play ending in some vertex $v \in V_{\mathbf{E}}$, a vertex v' such that $(v, v') \in E$. Éloïse *respects a strategy* Φ during some play $\Lambda = v_0 v_1 v_2 \cdots$ if $v_{i+1} = \Phi(v_0 \cdots v_i)$, for all $i \geq 0$ such that $v_i \in V_{\mathbf{E}}$. A strategy Φ for Éloïse is *winning* from some position $v \in V$ if she wins every play that starts from v and respects Φ. Finally, a vertex $v \in V$ is *winning* for Éloïse if she has a winning strategy from v, and the winning region for Éloïse consists of all winning vertices for her. Symmetrically, one defines the corresponding notions for Abelard.

A game $\mathbb{G}$ is *determined* if, from any position, either Éloïse or Abelard has a winning strategy. Determinacy of all games considered here follows from Martin's Theorem [30].

For more details and basic results on games, we refer the reader to [35, 40, 22, 39].

Higher-order pushdown processes. An *order-n pushdown process* is a tuple $\mathcal{P} = \langle Q, \Sigma, \bot, \delta \rangle$ where Q is a finite

set of control states, Σ is a finite stack alphabet containing a bottom-of-stack symbol $\perp \in \Sigma$, and Δ is a transition function (to be defined below). Let $Op_1 = \{pop_1, push_1(\sigma) \mid \sigma \in \Sigma \setminus \{\perp\}\}$ (resp. $Op_k = \{pop_k, push_k\}$) be the set of order-1 (resp. order-k with $2 \leq k \leq n$) stack operations. Then $\Delta : Q \times \Sigma \to 2^{Q \times Op_{\leq n}}$ where $Op_{\leq i} = \bigcup_{1 \leq k \leq i} Op_i$.

An *order*-0 *stack* over Σ is an element of Σ, and for $k > 0$ an *order-k stack* over Σ is a finite sequence $s_1, \cdots, s_\ell$ of order-$(k-1)$ stacks. Moreover we require that an order-k stack be non-empty whenever $k \geq 2$. Let $k > 0$ and take an order-k stack $s = s_1, \cdots, s_\ell$, we define the following partial functions:

$$top_1(s) = \begin{cases} s_\ell & \text{if } \ell \neq 0 \text{ and } k = 1, \\ top_1(s_\ell) & \text{if } k > 1. \end{cases}$$

$$pop_i(s) = \begin{cases} s_1, \cdots, s_{\ell-1} & \text{if } \ell > 1 \text{ and } k = i, \\ s_1, \cdots, s_{\ell-1}, pop_i(s_\ell) & \text{if } k > i. \end{cases}$$

$$push_i(s) = \begin{cases} s_1, s_2, \cdots, s_\ell, s_\ell & \text{if } k = i, \\ s_1, s_2 \cdots, s_{\ell-1}, push_i(s_\ell), & \text{if } k > i. \end{cases}$$

A configuration of $\mathcal{P}$ is a pair (q, s) where $q \in Q$ and s is an order-n stack. At a configuration (q, s), $\mathcal{P}$ can apply any transition $(q', op) \in \Delta(q, top_1(s))$, leading to the configuration $(q', op(s))$. The *configuration graph* of $\mathcal{P}$ is defined to be the graph whose vertices are the configurations and whose edges are given by the transitions of $\mathcal{P}$.

In order to define a notion of regularity over higher-order stacks, we associate to each order-k stack a well-bracketed word of depth k as follows. Let $s = s_1, \cdots, s_\ell$ be an order-k stack. We define $\widetilde{s} \in (\Sigma \cup \{[,]\})^*$ as

$$\widetilde{s} = \begin{cases} [\widetilde{s_1} \cdots \widetilde{s_\ell}] & \text{if } k \geq 1 \\ s & \text{if } k = 0 \text{ (\textit{i.e.} } s \in \Sigma) \end{cases}$$

A set S of order-k stacks is said to be *regular* if the language $\widetilde{S} = \{\widetilde{s} \mid s \in S\}$ is a regular subset of $(\Sigma \cup \{[,]\})^*$. By extension, a set C of configurations of an order-k pushdown automaton is said to be regular if the set $\widetilde{C} = \{\widetilde{s}q \mid (q, s) \in C\}$ is a regular subset of $(\Sigma \cup \{[,]\})^* Q$. For example $\{[\underbrace{[a\,a] \cdots [a\,a]}_{n}] \mid n \geq 0\}$ is a regular set of order-2 stacks, but $\{[\,[\underbrace{a \cdots a}_{n}]\,[\underbrace{a \cdots a}_{n}]\,] \mid n \geq 0\}$ is not.

In particular, the set of configurations reachable from a given configuration is not in general regular. Consider for example an order-2 pushdown automaton with two states q_0 and q_1 and two transitions $\Delta(q_0, a) = \{(q_0, push(a)), (q_1, push_2)\}$. The language of words representing the configurations reachable from $(q_0, [[a]])$ is the context-free language $\{[[a^n]]q_0 \mid n \geq 1\} \cup \{[[a^n][a^n]]q_1 \mid n \geq 1\}$.

When considering higher-order pushdown automata as acceptors of finite-word languages over a finite alphabet Σ, we attach to each transition a symbol in $\Sigma \cup \{\varepsilon\}$ and fix an initial state q_0 together with a set F of final states. We use the symbol ε to label silent transitions. We call such automata *labelled higher-order pushdown automata*.

A word $w \in \Sigma^*$ is accepted by the automaton if there exists a sequence of configurations $c_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} c_n$ where c_0 is the initial configuration $(q_0, [^n\perp]^n)$, c_n contains a final state in $F \subseteq Q$ and w is the word obtained by removing all occurrences of ε in $a_1 \ldots a_n$. (We say that w is the word *traced out* by the configuration sequence $c_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} c_n$.)

Proposition 1. *Take an (*resp. *deterministic) order-k pushdown automaton. The set of words traced out by configuration sequences from the initial configuration c_0 to a configuration c, where c ranges over a regular set of configurations, is accepted by an (*resp. *deterministic) order-k pushdown automaton.*

Abstract Pushdown Processes. We situate the techniques developed here in a general and abstract framework of (order-1) pushdown processes whose stack alphabet is a *possibly infinite* set.

An *abstract pushdown process* is a tuple $\mathcal{P} = \langle Q, \Gamma, \Delta \rangle$ where Q is a finite set of states, Γ is a (possibly infinite) set called an *abstract pushdown alphabet* and containing a bottom-of-stack symbol denoted $\perp \in \Gamma$, and

$$\Delta : Q \times \Gamma \to 2^{Q \times \{rew(\gamma), pop, push(\gamma) \mid \gamma \in \Gamma\}}$$

is the transition relation. We additionally require that for all $\gamma \neq \perp$, $\Delta(q, \gamma)$ does not contain any element of the form $(q, push(\perp))$ or $(q', rew(\perp))$, and that $\Delta(q, \perp)$ does not contain any element of the form (q', pop) or $(q', rew(\gamma))$ with $\gamma \neq \perp$, *i.e.* the bottom-of-stack symbol can only occur at the bottom of the stack, and is never popped or rewritten.

An *abstract pushdown content* is a word in $St = \perp(\Gamma \setminus \{\perp\})^*$. A configuration of $\mathcal{P}$ 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 rightmost symbol of σ.

Remark 1. In general an abstract pushdown process is not finitely describable, as the domain of Δ is infinite and no further assumption is made on Δ.

Example 1. A pushdown process is an abstract pushdown process whose stack alphabet is finite.

A abstract pushdown process $\mathcal{P}$ induces a possibly infinite graph, called an *abstract pushdown graph,* denoted $G = (V, E)$, whose vertices are the configurations of $\mathcal{P}$ (*i.e.* pairs from $Q \times St$), and edges are defined by the transition relation Δ, *i.e.*, from a vertex $(p, \sigma\gamma)$ one has edges to:

- $(q, \sigma\gamma')$ whenever $(q, rew(\gamma')) \in \Delta(p, \gamma)$.

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

Example 2. Higher-order pushdown processes are special cases of abstract pushdown processes. Let $n > 1$ and consider an order-n pushdown process $\mathcal{P} = \langle Q, \Sigma, \Delta \rangle$. Set Γ to be the set of all order-$(n-1)$ stacks over Σ, and for every $p \in Q$ and $\gamma \in \Gamma$ with $\sigma = top_1(\gamma)$, we define $\Delta'(p, \gamma)$ by

- $(q, pop) \in \Delta'(p, \gamma)$ iff $(q, pop_n) \in \Delta(q, \sigma)$;
- $(q, push(\gamma)) \in \Delta'(p, \gamma)$ iff $(q, push_n) \in \Delta(q, \sigma)$;
- $(q, rew(op(\gamma))) \in \Delta'(p, \gamma)$ iff $(q, op) \in \Delta(q, \sigma)$ where $k < n$ and op is an order-k action.

It follows that the abstract pushdown process $\langle Q, \Gamma, \Delta' \rangle$ and $\mathcal{P}$ have isomorphic transition graphs.

Abstract Pushdown Parity Game. Consider a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q between Éloïse and Abelard. 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 resulting game graph $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ is called an *abstract pushdown game graph*. Finally, an *abstract pushdown game* is a game played on such a game graph.

Let ρ be a colouring function from Q to a finite set of colours $C \subset \mathbb{N}$. This function is easily extended to a function from V to C by setting $\rho((q, \sigma)) = \rho(q)$. The parity condition is the winning condition defined by:

$$\Omega_{par} = \{v_0 v_1 \cdots \mid \liminf((\rho(v_i))_{i \geq 0}) \text{ is even}\}$$

For an abstract pushdown parity game, the main questions are the following:

1. For a given vertex, decide who wins from it.
2. For a given vertex, compute a winning strategy for the winning player.
3. Compute a finite representation of the winning regions.

These three problems are polynomially equivalent to the following problems respectively: model-checking for μ-calculus, controller synthesis against μ-calculus specifications, and global model-checking for μ-calculus.

Automata with oracles. We now define a class of automata to accept the winning positions in an abstract pushdown game. An *automaton with oracles* is a tuple $\mathcal{A} = (\mathcal{S}, Q, \Gamma, \delta, s_{\text{in}}, \mathcal{O}_1 \cdots \mathcal{O}_n, Acc)$ where $\mathcal{S}$ is a finite set of control states, Q is a set of input states, Γ is a (possibly infinite) input alphabet, $s_{\text{in}} \in S$ is the initial state, $\mathcal{O}_i$ are subsets of Γ (called *oracles*) and $\delta : \mathcal{S} \times \{0, 1\}^n \to \mathcal{S}$ is the transition function. Finally Acc is a function from $\mathcal{S}$ to 2^Q. Such an automaton is designed to accept in a *deterministic* way configurations of an abstract pushdown process whose abstract pushdown content alphabet is Γ and whose control states are Q.

Let $\mathcal{A} = (\mathcal{S}, Q, \Gamma, \delta, s_{\text{in}}, \mathcal{O}_1 \cdots \mathcal{O}_n, Acc)$ be such an automaton. With every $\gamma \in \Gamma$ we associate a Boolean vector $\pi(\gamma) = (b_1, \cdots b_n)$ where

$$b_i = \begin{cases} 1 & \text{if } \gamma \in \mathcal{O}_i \\ 0 & \text{otherwise.} \end{cases}$$

The automaton reads a configuration $C = (q, \gamma_1\gamma_2 \cdots \gamma_\ell)$ from left to right. A *run* over C is the sequence $s_0, \cdots, s_{\ell+1}$ such that $s_0 = s_{\text{in}}$ and $s_{i+1} = \delta(s_i, \pi(\gamma_i))$ for every $i = 0, \cdots, \ell$. Finally the run is *accepting* if and only if $q \in Acc(s_{\ell+1})$.

Remark 2. When the input alphabet is finite, it is easily seen that automata with oracles behave as (standard) deterministic finite automata.

In this article, we are going to use automata with oracles to accept sets of configurations of higher-order pushdown automata. As seen in Example 2 for an order-$(k+1)$ pushdown automaton, we take Γ to be the set of all order-k stacks. The sets of regular configurations of an order-$(k+1)$ pushdown automaton are naturally captured by automata using, as oracles, regular sets of order-k stacks.

Proposition 2. *Fix an order-$(k+1)$ pushdown automaton $\mathcal{P}$ and consider an automaton $\mathcal{A}$ with oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$ respectively accepted by deterministic word automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Let C be the set of configurations of $\mathcal{P}$ accepted by $\mathcal{A}$. Then we can construct a deterministic finite automaton, of size $\mathcal{O}(|\mathcal{A}||\mathcal{A}_1| \cdots |\mathcal{A}_n|)$, accepting the set $\widetilde{C}$.*

3 Preliminary results

From now on, let us fix an abstract pushdown process $\mathcal{P} = \langle Q, \Gamma, \Delta \rangle$ together with a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function ρ using a finite set of colours C. Denote respectively by $\mathcal{G} = (V, E)$ and $\mathbb{G}$ the associated abstract pushdown game graph and abstract pushdown parity game.

We can define an automaton with oracles that accepts Éloïse's winning region of the game $\mathbb{G}$. The oracles of this automaton are defined using conditional games. For every subset R of Q the game $\mathbb{G}(R)$ played over $\mathcal{G}$ is the *conditional game induced by R over $\mathcal{G}$*. A play Λ in $\mathbb{G}(R)$ is winning for Éloïse iff one of the following happens:

- In Λ no configuration with an empty stack (*i.e.* of the form $(q, \bot)$) is visited, and Λ satisfies the parity condition.

- In Λ a configuration with an empty stack is visited and the control state in the first such configuration belongs to R.

More formally, the winning condition in $\mathbb{G}(R)$ is

$$[\Omega_{par} \setminus V^*(Q \times \{\bot\})V^\omega] \cup V^*(R \times \{\bot\})V^\omega$$

For any state q, any stack letter $\gamma \neq \bot$, and any subset $R \subseteq Q$ it follows from Martin's Determinacy theorem that either Éloïse or Abelard has a winning strategy from $(q, \bot\gamma)$ in $\mathbb{G}(R)$. We denote by $\mathcal{R}(q,\gamma)$ the set of subsets R for which Éloïse wins in $\mathbb{G}(R)$ from $(q, \bot\gamma)$:

$$\mathcal{R}(q,\gamma) = \{R \subseteq Q \mid (q, \bot\gamma) \text{ is winning for Éloïse in } \mathbb{G}(R)\}$$

Proposition 3. *Let $\sigma \in (\Gamma\setminus\{\bot\})^*$, $q \in Q$ and $\gamma \in \Gamma\setminus\{\bot\}$. Then Éloïse has a winning strategy in $\mathbb{G}$ from $(q, \bot\sigma\gamma)$ if and only if there exists some $R \in \mathcal{R}(q,\gamma)$ such that $(r, \bot\sigma)$ is winning for Éloïse in $\mathbb{G}$ for every $r \in R$.*

Proof (sketch). Assume Éloïse has a winning strategy from $(q, \bot\sigma\gamma)$ in $\mathbb{G}$ and call it φ. Define R to be the set of all $r \in Q$ such that there is a play $v_0 \cdots v_k(r, \bot\sigma)v_{k+1}\cdots$ where Éloïse respects φ and each v_i for $0 \leq i \leq k$ is of the form $(p, \bot\sigma\sigma')$ for some $\sigma' \neq \varepsilon$. Mimicking φ, Éloïse wins in $\mathbb{G}(R)$ from (q,γ) and hence $R \in \mathcal{R}(q,\gamma)$. Finally, for every $r \in R$ there is a partial play λ_r that starts from $(q, \bot\sigma\gamma)$, where Éloïse respects φ, and that ends in $(r, \bot\sigma)$. Hence $(r, \bot\sigma)$ is also winning for Éloïse in $\mathbb{G}$.

Conversely, let us assume that there is some $R \in \mathcal{R}(q,\gamma)$ such that $(r, \bot\sigma)$ is winning for Éloïse in $\mathbb{G}$ for every $r \in R$. For every $r \in R$, let us denote by φ_r a winning strategy for Éloïse from $(r, \bot\sigma)$ in $\mathbb{G}$. Let φ_R be a winning strategy for Éloïse in $\mathbb{G}(R)$ from $(q, \bot\gamma)$. In order to win in $\mathbb{G}$ from $(q, \bot\sigma\gamma)$, Éloïse first mimics φ_R and, if eventually some configuration $(r, \bot\sigma)$ is reached she follows, φ_r from that point onward. □

Proposition 3 easily implies the following result.

Theorem 1. *Let $\mathbb{G}$ be an abstract pushdown parity game induced by an abstract pushdown process $\mathcal{P} = \langle Q, \Gamma, \Delta\rangle$. Then the set of winning positions in $\mathbb{G}$ for Éloïse (respectively for Abelard) is accepted by an automaton with oracles $\mathcal{A} = (\mathcal{S}, Q, \Gamma, \delta, s_i, \mathcal{O}_1 \cdots \mathcal{O}_n, Acc)$ such that*

- $\mathcal{S} = 2^Q$;
- $s_i = \emptyset$.
- *There is an oracle $\mathcal{O}_{p,R}$ for every $p \in Q$ and $R \subseteq Q$, and $\gamma \in \mathcal{O}_{p,R}$ iff $R \in \mathcal{R}(p,\gamma)$ and $\gamma \neq \bot$.*
- *There is an oracle $\mathcal{O}_\bot$ and $\gamma \in \mathcal{O}_\bot$ iff $\gamma = \bot$*
- *Using the oracles, δ is designed such that:*
  - *From state $\emptyset$ on reading $\bot$, $\mathcal{A}$ goes to $\{p \mid (p, \bot)$ is winning for Éloïse in $\mathbb{G}\}$.*
  - *From state S on reading γ, $\mathcal{A}$ goes to $\{p \mid S \in \mathcal{R}(p,\gamma)\}$.*
- *Acc is the identity function.*

We will later use Theorem 1 in combination with Remark 2 to prove that the set of winning positions in any higher-order pushdown parity games is regular (see Theorem 4).

4 Reducing the conditional games

The main purpose of this section is to build a new game whose winning region embeds all the information needed to determine the sets $\mathcal{R}(q,\gamma)$. Moreover the underlying game graph no longer uses a stack.

For an infinite play $\Lambda = v_0 v_1 \cdots$, let $Steps_\Lambda$ be the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally, $Steps_\Lambda = \{i \in \mathbb{N} \mid \forall j \geq i\ sh(v_j) \geq sh(v_i)\}$, where $sh((q, \bot\gamma_1 \cdots \gamma_n)) = n+1$. Note that $Steps_\Lambda$ is always infinite and hence induces a factorisation of the play Λ into finite pieces.

In the factorisation induced by $Steps_\Lambda$, a factor $v_i \cdots v_j$ is called a *bump* if $sh(v_j) = sh(v_i)$, called a *Stair* otherwise (that is, if $sh(v_j) = sh(v_i) + 1$).

For any play Λ with $Steps_\Lambda = \{n_0 < n_1 < \cdots\}$, we can define the sequence $(mcol_i^\Lambda)_{i\geq 0} \in \mathbb{N}^\mathbb{N}$ by setting $mcol_i^\Lambda = \min\{\rho(v_k) \mid n_i \leq k \leq n_{i+1}\}$. This sequence fully characterises the parity condition.

Proposition 4. *For a play Λ, $\Lambda \in \Omega_{par}$ iff $\liminf((mcol_i^\Lambda)_{i\geq 0})$ is even.*

In the sequel, we build a new parity game $\widetilde{\mathbb{G}}$ over a new game graph $\widetilde{\mathcal{G}} = (\widetilde{V}, \widetilde{E})$. This new game *simulates* the abstract pushdown graph, in the sense that the sequence of visited colours during a correct simulation of some play Λ in $\mathbb{G}$ is exactly the sequence $(mcol_i^\Lambda)_{i\geq 0}$. Moreover, a play in which a player does not correctly simulate the abstract pushdown game is losing for that player. We shall see that the winning regions in $\widetilde{\mathbb{G}}$ allow us to compute the sets $\{\gamma \in \Gamma \mid R \in \mathcal{R}(q,\gamma)\}$.

Before providing a description of the game graph $\widetilde{\mathcal{G}}$, let us consider the following informal description of this simulation game. We aim at simulating a play in the abstract pushdown game from some initial vertex $(p_{in}, \bot)$. In $\widetilde{\mathcal{G}}$ we keep track of only the control state and the top stack symbol of the simulated configuration.

The interesting case is when it is in a control state p with top stack symbol α, and the player owning p wants to push a symbol β onto the stack and change the control state to q.

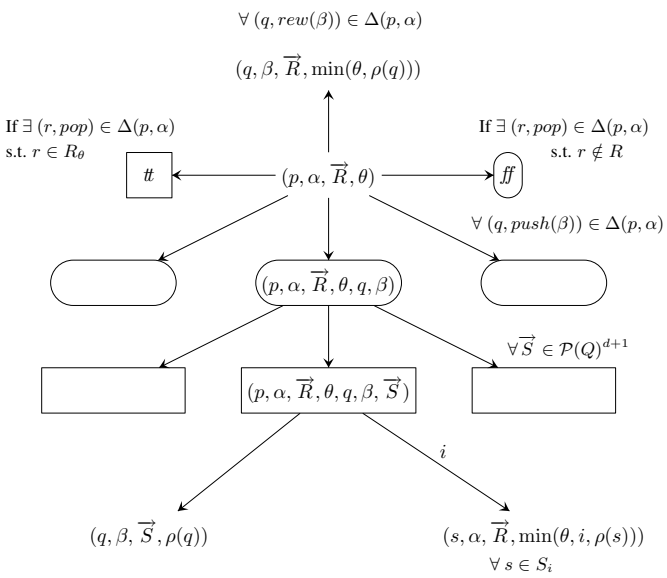

Figure 1. Local structure of $\widetilde{\mathcal{G}}$.

For every strategy of Éloïse, there is a certain set of possible (finite) continuations of the play that will end with popping β (or actually a symbol into which β was rewritten in the meantime) from the stack. We require Éloïse to declare a vector $\overrightarrow{S} = (S_0, \ldots, S_d)$ of $(d+1)$ subsets of Q, where S_i is the set of all states the game can be in after popping (possibly a rewriting of) β along those plays where in addition the smallest visited colour while (possibly a rewriting of) β was on the stack is i.

Abelard has two choices. He can continue the game by pushing β onto the stack and updating the state (we call this a *pursue* move). Otherwise, he can pick a set S_i and a state $s \in S_i$, and continue the simulation from that state s (we call this a *jump* move). If he does a pursue move, then he remembers the vector $\overrightarrow{S}$ claimed by Éloïse; if later on, a pop transition is simulated, the play stops and Éloïse wins if and only if the resulting state is in S_θ where θ is the smallest colour seen in the current level (this information is encoded in the control state, reset after each pursue move and updated after each jump move). If Abelard does a jump move to a state s in S_i, the currently stored value for θ is updated to $\min(\theta, i, \rho(s))$, which is the smallest colour seen since the current stack level was reached.

There are extra moves to simulate rew rules where the top stack element and the value of θ are updated.

Therefore the main vertices of this new game graph are of the form $(p, \alpha, \overrightarrow{R}, \theta)$, which are controlled by the player who controls p. Intermediate vertices are used to handle the previously described intermediate steps. The local structure is given in Figure 1 (circled vertices are those controlled by Éloïse). Two special vertices *#* and *ff* are used to simulate pop moves. This game graph is equipped with a colouring function on the vertices and on the edges: vertices of the form $(p, \alpha, \overrightarrow{R}, \theta)$ have colour $\rho(p)$, an edge leaving from a vertex $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$ has colour i where i is the colour of the simulated bump. Note that intermediate vertices could be introduced in order to have only colours on vertices. A precise description of the graph is given in the full proof of the following main result.

Theorem 2. *The following holds.*

1. *A configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}$ if and only if $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \rho(p_{in}))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
2. *For every $q \in Q$, $\gamma \in \Gamma$ and $R \subseteq Q$, $R \in \mathcal{R}(q, \gamma)$ if and only if $(q, \gamma, (R, \ldots, R), \rho(q))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*

Proof (sketch). We concentrate here on the first point, as the second is actually a part of the proof of the first one. For the direct implication, assume that the configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}$, and let Φ be a corresponding strategy for her.

Using Φ, we define a strategy φ for Éloïse in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \rho(p_{in}))$. This strategy stores a partial play Λ in $\mathbb{G}$, that is an element in V^* (where V denotes the set of vertices of $\mathcal{G}$). At the beginning Λ is initialised to the vertex $(p_{in}, \bot)$.

By a *round* we mean a factor of a play between two visits through vertices of the form $(p, \alpha, \overrightarrow{R}, \theta)$. Both the strategy φ and the update of Λ, are described for a round. When Éloïse has to play from some vertex $(p, \alpha, \overrightarrow{R}, \theta)$ she considers the value of $\Phi(\Lambda)$. If it is a pop move then she goes to *#* (one proves that this move is always possible). If it equals $(q, rew(\beta))$, she goes to $(q, \beta, \overrightarrow{R}, \min(\theta, \rho(q)))$. Finally, if it is equal to $(q, push(\beta))$, she goes to $(p, \alpha, \overrightarrow{R}, \theta, \beta)$.

From some vertex $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$, Éloïse has to provide a vector $\overrightarrow{S} \in \mathcal{P}(Q)^{d+1}$ that describes which states can be reached if β (or one of its successors by top rewriting) is popped, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{S}$, Éloïse considers the set of all possible continuations of $\Lambda \cdot (q, \sigma\alpha\beta)$ (where $(p, \sigma\alpha)$ denotes the last vertex of Λ) where she respects her strategy Φ. For each such play, she checks whether some configuration of the form $(s, \sigma\alpha)$ is visited after $\Lambda \cdot (q, \sigma\alpha\beta)$, that is if the β is eventually popped. If it is the case, she considers the first such configuration and the smallest colour i seen in the meantime. For every $i \in \{0, \ldots d\}$, S_i, is exactly the set of states $s \in Q$ such that the preceding case happens.

Let $(p, \sigma\alpha)$ be the last vertex in Λ. The memory Λ is updated after each visit to a vertex of the form $(p, \alpha, \overrightarrow{R}, \theta)$,

we have three cases depending on the kind of the round. If it was simulating a $(q, rew(\beta))$ action then the updated memory is $\Lambda \cdot (q, \beta\sigma)$. If it was simulating a bump of colour i starting with some action $(q, push(\beta))$ and ending in a state $s \in S_i$ then the memory becomes Λ extended by $(q, \sigma\alpha\beta)$ followed by a sequence of moves, where Éloïse respects Φ, that ends by popping β and reaches $(s, \sigma\alpha)$ while having i as smallest colour. Finally, if it was simulating a stair starting with a $(q, push(\beta))$ action, then the updated memory is $\Lambda \cdot (q, \sigma\alpha\beta)$.

Therefore, any partial play λ in $\widetilde{\mathbb{G}}$ — in which Eloise respects her strategy φ — is associated with a partial play Λ in $\mathbb{G}$. One shows that Éloïse respects Φ in Λ. The same arguments work for an infinite play λ, and the corresponding play Λ is infinite, starts from $(p_{in}, \bot)$ and Éloïse respects Φ in that play. Therefore it is a winning play. Relying on that fact one concludes that φ is winning.

For the converse implication one can reason in a rather similar way by constructing a winning strategy for Abelard in $\widetilde{\mathbb{G}}$ from one in $\mathbb{G}$. □

From a more constructive proof of Theorem 2 one can construct natural strategies in $\mathbb{G}$ from strategies in $\widetilde{\mathbb{G}}$.

Theorem 3. *Assume Éloïse has a winning strategy φ in $\widetilde{\mathbb{G}}$ that uses a memory ranging from some set M. Then one can construct an abstract pushdown process $\mathcal{T}$ with output that realises a winning strategy Φ for Éloïse in $\mathbb{G}$. Moreover the abstract pushdown alphabet used by $\mathcal{T}$ is $\widetilde{V} \times M$ and, at any moment in a play where Éloïse respects Φ, the abstract pushdown content of $\mathcal{T}$ has exactly the same height as the one in the current position of the game graph. Finally, if φ is effective the same holds for Φ.*

Remark 3. In the special case of pushdown games, since Γ is finite so is $\widetilde{\mathbb{G}}$. Hence in the previous statement, φ can be chosen to be memoryless. Therefore one concludes that for pushdown games one can construct a deterministic pushdown automaton that realises a winning strategy and whose stack is synchronised with the one in the game [38].

5 Uniform solution of higher-order pushdown parity games and strategies

In this section we prove that the winning regions in higher-order pushdown games are regular. The first step is to note the following property.

Property 1. *Let $\mathbb{G}$ be a higher-order pushdown parity game of order-n and let $\widetilde{\mathbb{G}}$ be as in Theorem 2. Then $\widetilde{\mathbb{G}}$ is a higher-order pushdown parity game of order-$(n-1)$.*

Proof. One simply needs to consider how the game graph $\widetilde{\mathcal{G}}$ is defined. It suffices to make the following observations concerning the local structure given in Figure 1 when $\mathbb{G}$ is played on the transition graph of a pushdown automaton of order-n.

1. For every vertex of the form $(p, \alpha, \overrightarrow{R}, \theta)$, $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$ or $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$, α is an order-$(n-1)$ stack.
2. For every vertex of the form $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$ or $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$, it holds that $\alpha = \beta$.

This implies that any vertex in $\widetilde{\mathcal{G}}$ can be seen as a pair formed by a state in a finite set and an order-$(n-1)$ stack. Then one concludes the proof by checking that the edge relation is the one of an order-$(n-1)$ pushdown automaton (for the transition to vertices *tt* and *ff* one can introduce vertices (tt, α) and (ff, α) for any order-$(n-1)$ stack α). □

Remark 4. The number states of the higher-order pushdown automaton describing $\widetilde{\mathbb{G}}$ is exponential in the number of states of the pushdown automaton describing $\mathbb{G}$ but both games have the same number of colours.

Consider the order-1 case. As $\widetilde{\mathbb{G}}$ is finite one can solve it and therefore effectively construct the automaton with oracles as in Theorem 1. As this automaton has a finite input alphabet, using Remark 2, we deduce the following result.

Property 2. *[7, 32] The set of winning position in a pushdown parity game is regular.*

Now, one can iterate this reasoning: applying inductively Property 1 together with Proposition 2 and Theorem 2 easily leads to the desired result.

Theorem 4. *The sets of winning positions in a higher-order pushdown parity game are regular and can be effectively computed. Computing these regions is an n-ExpTime-complete problem for an order-n pushdown parity game.*

Starting with an order-n pushdown parity game, and applying n times the reduction of Proposition 1, one ends up with a parity game using the same number of colours over a finite game graph whose size, using Remark 4, is n times exponential in the size of the original order-n pushdown automaton. As solving this latter game is exponential only on the number of colours [22] the global procedure is in n-ExpTime. The lower bound follows from the fact that deciding the winner in two-player reachability games over order-n pushdown is already n-ExpTime-hard. A self-contained proof can be found in [9]. The next remark sketches a much simpler proof of this result.

Remark 5. Note that using the reduction of Theorem 2, we can deduce n-ExpTime-hardness by reduction to the $(n-1)$-ExpTime-hardness of the emptiness problem for

order-n pushdown automata [17]. Consider an order-$(n+1)$ pushdown automaton $\mathcal{P}$ over a one-letter input alphabet. The emptiness problem for $\mathcal{P}$ is polynomially equivalent to deciding the winner in the associated one-player reachability game. In the case of a one-player game, the reduction of Theorem 2 can be tailored to yield a reduced game $\widetilde{\mathbb{G}}$ which is a two-player game of *polynomial* size w.r.t to $\mathcal{P}$. Hence we establish that the emptiness problem for order-$(n+1)$ pushdown automata can be polynomially reduced to deciding the winner of two-player order-n game. This proves the n-ExpTime-hardness of the latter problem.

Remark 6. Theorem 4 generalises the result obtained by Hague and Ong [25] for higher-order pushdown reachability games.

Concerning strategies, we already noted in Remark 3 that a winning strategy can be realised by a pushdown automaton whose stack is synchronised with the one from the game. Reasoning by induction and relying on Theorem 3, one can obtain a similar result for the general case. More precisely, if one defines the shape of an order-k stack to be the stack of obtained by rewriting the stack symbols by a fixed symbol $\diamond$, we get the following result.

Theorem 5. *Consider an order-k pushdown parity game. Then one can construct, for any player, a deterministic order-k pushdown automaton that realises a winning strategy and whose stack has always the same shape as the one in the game.*

Remark 7. Theorem 5 means that the memory needed to win a higher-order pushdown parity game can actually be implemented in the underlying higher-order pushdown automaton defining the game by enriching its stack alphabet.

6 Global μ-calculus model-checking

Given a vertex (state) s in a state-transition graph $\mathcal{K}$ and a formula φ, the *model-checking problem* asks whether $\mathcal{K}, s \models \varphi$ holds (in words, whether the formula φ holds at s in the structure $\mathcal{K}$). The *global model-checking problem* is the task of computing (if possible) a finite representation of the set $||\mathcal{K}||_\varphi = \{s \mid \mathcal{K}, s \models \varphi\}$ of vertices in $\mathcal{K}$ that satisfy a given formula φ.

In the following we tackle the global model checking problem when φ is a formula of the modal μ-calculus [29, 2]. We consider two kinds of structures for $\mathcal{K}$: configuration graphs of higher-order pushdown automata, and trees generated by higher-order *safe* recursion schemes.

Global model-checking of configuration graphs of higher-order pushdown automata

The μ-calculus global model-checking problem for families of graphs closed under Cartesian product with finite graphs is well-known to be equivalent to the solvability of parity games over the same class. Hence Theorem 4 implies the following characterisation of μ-calculus definable sets over higher-order pushdown graphs:

Theorem 6. *The μ-calculus definable sets over configuration graphs of higher-order pushdown automata are exactly the regular sets of configurations.*

Remark 8. The preceding Theorem generalises a result of Bouajjani and Meyer in [4] for a weaker logic (*i.e.* the $E(U, X)$ fragment of CTL) over weaker structures (*i.e.* higher-order pushdown automata with a single control state).

Global model-checking of trees generated by higher-order safe recursion schemes

Fix a (ranked) alphabet Σ. *Types* are generated from a base type using the arrow constructor $\rightarrow$. A higher-order (deterministic) recursion scheme is a finite set of equations of the form $F\, x_1 \cdots x_n = e$, where F is a typed non-terminal, each x_i is a typed variable, and e is an applicative term constructed from the non-terminals, terminals (which are the Σ-symbols), and variables $x_1, \cdots, x_n$. The scheme is said to be *order-k* if the highest order of the non-terminals is k. We use recursion schemes here as generators of possibly infinite term-trees. The ranked *tree* generated by a recursion scheme is defined to be the (possibly infinite) term-tree built up from the terminal symbols by applying the equations *qua* rewrite rules, replacing the formal parameters by the actual parameters, starting from the initial non-terminal. We refer the reader to [28] for a precise description of the preceding, and for the meaning of *safety* which is a syntactic constraint; here we rely on the following characterisation:

Theorem 7. *[28] For each $k \geq 0$, the ranked trees generated by safe order-k recursion schemes coincide with the ε-closure of the unravelling of the configuration graphs of deterministic order-k pushdown automata.*

Consider a higher-order deterministic pushdown automaton labelled by $\Sigma \cup \{\varepsilon\}$. After unravelling its configuration graph from the initial configuration, the operation of *ε-closure* (see [34]) first adds an a-labelled edge from c_1 to c_2 whenever there is a path from c_1 to c_2 that traces out the word $a\varepsilon^*$ and c_2 is not the source-vertex of an ε-labelled edge, and then removes the source-vertex of every ε-labelled edge. The resultant graph is a tree.

A node in this tree is naturally represented by the word over Σ labelling the path from the root to this node. Hence, every μ-calculus formula over such a tree induces a language in Σ^* *i.e.* the set of words labelling a path from the root to a node satisfying the formula. These languages can be characterised as follows:

Theorem 8. *Let $k \geq 0$ and let $\mathcal{T}$ be the ranked tree generated by a given order-k safe recursion scheme. The μ-calculus definable sets of $\mathcal{T}$-nodes are recognisable by a deterministic order-k pushdown automaton.*

Proof (sketch). Let $\mathcal{C}$ be the configuration graph of a deterministic order-k $\Sigma \cup \{\varepsilon\}$-labelled pushdown automaton $\mathcal{P}$, and let U be the unravelling of $\mathcal{C}$ from the initial configuration c_0, such that the ε-closure of U is isomorphic to $\mathcal{T}$. Let φ be a μ-calculus formula. A node in U satisfies φ if and only if it corresponds to a path ending in a vertex of $\mathcal{C}$ that satisfies φ. By Theorem 6, the set F of configurations of $\mathcal{C}$ satisfying φ in $\mathcal{C}$ is regular. Moreover it is easy to see that the set E of configurations of $\mathcal{C}$ which are not the source-vertex of a ε-labelled edge in $\mathcal{C}$ is also regular. A node in $\mathcal{T}$ at the end of a path from the root labelled by w satisfies φ in $\mathcal{T}$ if and only if w is accepted by the labelled pushdown automaton $\mathcal{P}$ from the initial configuration to the regular set $E \cap F$. By Proposition 1, the language consisting of such w is accepted by an order-k pushdown automaton. □

7 Discussions

There are a number of further directions:

• Can the quite general class of stack data games together with Theorem 2 be used to prove the decidability of games over new structures (*e.g.* allowing arithmetic on the stack)?

• Regular stack properties allow assertions over the stack contents. Since information regarding regular tests may be encoded in the control states and stack of an abstract pushdown process, the definable sets are again regular. Properties of this kind have been shown to have applications in inter-procedural data-flow analysis [18] and security [27] for the case of order-1 pushdown systems. One such security property, implemented in Java and .Net [21, 5], allows the programmer to decorate code with permission checks. These checks require that all callers on the stack have sufficient privileges to proceed. Do regular stack properties have similar applications for other structures encodable as abstract pushdown processes?

• The notion of regularity used in this article can capture the μ-calculus definable sets of configurations. As mentioned previously, MSO-definable sets can be captured by a stronger notion of regularity introduced in [10, 20]. Hence a natural question is the decidability of the following problem: given a strong regular set (in the sense of [10, 20]), decide whether it is regular (in the sense of this article). Moreover, it is known from [20] that positional winning strategies of higher-order pushdown parity game can be described using strong regular sets (*i.e.* for each transition, the set of configurations from which the strategy plays the transition is a strong regular set) and a k-Exptime algorithm was recently given in [12]. Is it possible to describe positional winning strategies using the weak notion of regularity of this article?

• *Order-k collapsible pushdown automata* (CPDA) are a generalisation of order-k pushdown automata in which each symbol in the k-stack has a link to a stack below it. As generators of ranked trees, they are equi-expressive with (arbitrary) order-k recursion schemes [24]. A natural question is to consider configuration graphs of CPDA, and compute a finite representation of the μ-calculus definable vertex-sets thereof.

An alternative method for computing the winning regions of a higher-order pushdown parity game develops the order-1 saturation techniques introduced by Bouajjani *et al.* [3] and Finkel *et al.* [19] and generalised to Büchi games by Cachat [8]. This algorithm uses a characterisation of a game's winning regions as a series of greatest and least fixed points. Following this characterisation, a small initial automaton, accepting higher-order stacks, is expanded until the winning region has been computed. Because this technique does not require an n-exponential reduction to a finite state game, it is possible that the state-explosion problem may be avoided in some, low-order, cases.

References

[1] A. Aho. Indexed grammars, an extension of context-free grammars. *JACM*, 15:647-671, 1968.

[2] A. Arnold and D. Niwiński. *Rudiments of mu-calculus*, 2001.

[3] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. CONCUR'97*, 1997.

[4] A. Bouajjani and A. Meyer. Symbolic reachability analysis of higher-order context-free processes. In *Proc. FSTTCS'04*, pp 135-147, 2004.

[5] D. Box and T. Pattison. *Essential .NET: The Common Language Runtime*. Addison-Wesley Longman, 2002.

[6] T. Cachat. Uniform solution of parity games on prefix-recognizable graphs. In *Proc. INFINITY'02*, 2002.

[7] T. Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *Proc. ICALP'03*, 2003.

[8] T. Cachat. *Games on Pushdown Graphs and Extensions*. PhD thesis, RWTH Aachen, 2003.

[9] T. Cachat and I. Walukiewicz. The complexity of games on higher order pushdown automata, 2007.

[10] A. Carayol. Regular sets of higher-order pushdown stacks. In *Proc. MFCS'05*, pp 168-179, 2005.

[11] A. Carayol, M. Hague, A. Meyer, C.-H. L. Ong, and O. Serre. Winning regions of higher-order pushdown games. Tech. Report, 2007. 20 pages, downloable from `http://www.liafa.jussieu.fr/~serre/`.

[12] A. Carayol and M. Slaats. Positional Strategies for Higher-Order Pushdown Parity Games. Submitted, 2008.

[13] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proc. FSTTCS'03*, pp 112-123, 2003.

[14] D. Caucal. On infinite terms having a decidable monadic theory. In *Proc. MFCS'02*, pp 165-176, 2002.

[15] W. Damm. The IO- and OI-hierarchies. *TCS*, 20:95-207, 1982.

[16] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. FoCS'91*, pp 368-377, 1991.

[17] J. Engelfriet. Iterated stack automata and complexity classes. *Info. & Comp.* 95:21-75, 1991.

[18] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. In *Proc. TACAS'01*, pp 306-339, 2001.

[19] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. In *Proc. INFINITY'97*, 1997.

[20] S. Fratani. *Automates à pile de piles ... de piles*. PhD thesis, Université de Bordeaux I, 2005.

[21] L. Gong. *Inside Java 2 platform security architecture, API design, and implementation*. Addison-Wesley Longman, 1999.

[22] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, LNCS 2500, 2002.

[23] M. Hague. *Global Model Checking of Higher-Order Pushdown Systems*. PhD thesis, Oxford Univ., 2008.

[24] M. Hague, A. Murawski, C.-H. L. Ong and O. Serre. Collapsible pushdown automata and recursion schemes. In *Proc. LICS'08*, to appear.

[25] M. Hague and C.-H. L. Ong. Symbolic backwards-reachability analysis for higher-order pushdown systems. In *Proc. FOSSACS'07*, pp 213-227, 2007.

[26] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *Proc. CONCUR'96*, pp 263-277, 1996.

[27] T. P. Jensen, D. Le Métayer, and T. Thorn. Verification of control flow based security properties. In *IEEE Symp. Security and Privacy*, pp 89-103, 1999.

[28] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proc. FOSSACS'02*, pp 205-222, 2002.

[29] D. Kozen. Results on the propositional mu-calculus. *TCS*, 27:333-354, 1983.

[30] D. A. Martin. Borel determinacy. *Annals of Maths.*, 102: 363-371, 1975.

[31] A. N. Maslov. Multi-level stack automata. *Problems Information Transmission*, 12:38-43, 1976.

[32] O. Serre. Note on winning positions on pushdown games with omega-regular winning conditions. *IPL*, 85:285-291, 2003.

[33] O. Serre. Games with winning conditions of high borel complexity. *TCS*, 350(2-3):345-372, 2006.

[34] C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Technical Report EDI-INF-RR-0005, University of Edinburgh, 2000.

[35] W. Thomas. On the synthesis of strategies in infinite games. In *Proc. STACS'95*, pp 1-13, 1995.

[36] W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In *Proc. MFCS'03*, pp 113-124, 2003.

[37] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. ICALP'98*, pp 628-641, 1998.

[38] I. Walukiewicz. Pushdown processes: games and model checking. *Info. & Comp.*, 157:234-263, 2000.

[39] I. Walukiewicz. A landscape with games in the backgroung. In *Proc. LICS'04*, pp 356-366, 2004.

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

Appendix

A Proof of Proposition 1

Proposition 1. *Take an (*resp. *deterministic) order-k pushdown automaton. The set of words traced out by configuration sequences from the initial configuration c_0 to a configuration c, where c ranges over a regular set of configurations, is accepted by an (*resp. *deterministic) order-k pushdown automaton.*

Proof. Let $\mathcal{P} = (Q, \Sigma, \Delta_{\mathcal{P}})$ be an order-$(k+1)$ pushdown automaton. Let Γ be the set of all order-k stacks over Σ.

Let $\mathcal{A} = (Q_{\mathcal{A}}, \delta_{\mathcal{A}}, q_0^{\mathcal{A}}, F_{\mathcal{A}})$ be a deterministic automaton over the alphabet $\Sigma \cup Q \cup \{[,]\}$ accepting words of the form wq with w a well-bracketed word of depth $k+1$ over $\Sigma \cup \{[,]\}$ and $q \in Q$.

Let Θ be the set of partial functions from $Q_{\mathcal{A}}$ to $Q_{\mathcal{A}}$. Moreover with every word $w \in \Sigma \cup \{[,]\}$, we associate the partial function $\theta_w \in \Theta$ defined by $\theta_w(q) = \delta_{\mathcal{A}}(q, w)$ for $q \in Q_{\mathcal{A}}$.

For an order-k stack $s = (s_1, \ldots, s_{|s|})$, we take for all $i \in [1, |s|]$, $\theta_i^s \in \Theta$ to be $\theta_{\tilde{s_1} \cdots \tilde{s_i}}$.

With every order-k stack s over Σ we associate an order-k stack $[\![s]\!]$ over $\Sigma \cup \Theta$ as follows.

For an order-1 stack $s = (\gamma_1, \ldots, \gamma_\ell)$, we take $[\![s]\!] := (\gamma_1, \theta_1^s, \ldots, \theta_{\ell-1}^s, \gamma_\ell)$. For an order-$(k+1)$ stack $s = (s_1, \ldots, s_\ell)$, we take $[\![s]\!] := (s'_1, \ldots, s'_\ell)$ for all $i \in [1, \ell]$, where s'_i is obtained by pushing the symbol θ_i^s on top of $[\![s_i]\!]$.

The idea of the construction is to simulate a configuration (q, s) of the original automaton by the configuration $((q, \gamma, \theta_1, \ldots, \theta_k), [\![s]\!])$ where γ is the top-most symbol of s and where for all $i \in [1, k]$, $\theta_i = \theta_{|t_i|-1}^{t_i}$ where t_i is the top-most order-i stack of s.

It is easy to check that a transition of the original automaton can be similuated in a deterministic fashion by a finite sequence of operations. To process sequences of operations, we naturally introduce intermediate states and label the corresponding transition by the silent symbol ε.

Finally a state $(q, \gamma, \theta_1, \ldots, \theta_k)$ is final if $\delta_{\mathcal{A}}(\overrightarrow{\theta_1} \ldots \overrightarrow{\theta_k}(q_0^{\mathcal{A}}), \gamma]^k q) \in F_{\mathcal{A}}$ where $\overrightarrow{\theta_i}(q) = \theta_i(\delta(q, [))$. □

B Proof of Proposition 2

Proposition 2. *Fix an order-$(k+1)$ pushdown automaton $\mathcal{P}$ and consider an automaton $\mathcal{A}$ with oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$ respectively accepted by deterministic word automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Let C be the set of configurations of $\mathcal{P}$ accepted by $\mathcal{A}$. Then we can construct a deterministic finite automaton, of size $\mathcal{O}(|\mathcal{A}||\mathcal{A}_1| \cdots |\mathcal{A}_n|)$, accepting the set $\widetilde{C}$.*

Proof. Let $\mathcal{P} = (Q, \Sigma, \Delta_{\mathcal{P}})$ be an order-$(k+1)$ pushdown automaton and let Γ be the set of all order-k stacks over Σ.

For the direct implication, consider an automaton with oracles $\mathcal{A} = (\mathcal{S}, Q, \Gamma, \delta, s_{\text{in}}, \mathcal{O}_1, \ldots, \mathcal{O}_n, Acc)$. For $i \in [1, n]$, we assumme that $\mathcal{O}_i \setminus \{\bot\}$ is accepted by the deterministic finite automaton $\mathcal{A}_i = (Q_i, \delta_i, q_0^i, F_i)$ over the alphabet $\Sigma \cup \{[,]\}$.

We construct a determistic finite automaton $\mathcal{B} = (Q_{\mathcal{B}}, \delta_{\mathcal{B}}, q_0^{\mathcal{B}}, F_{\mathcal{B}})$ over $\Sigma \cup Q \cup \{[,]\}$ such that for every order-$(k+1)$ stack $(\gamma_1, \ldots, \gamma_\ell)$ and every state $q \in Q$, $((\bot, \gamma_1, \ldots, \gamma_\ell), q)$ is accepted by $\mathcal{A}$ if and only if the word $[\tilde{\gamma_1} \cdots \tilde{\gamma_\ell}]q$ is accepted by $\mathcal{B}$.

The set of states $Q_{\mathcal{B}}$ of $\mathcal{B}$ is equal to $\mathcal{S} \times Q_1 \times \ldots \times Q_n \times [0, n] \cup \{q_0^{\mathcal{B}}, q_f^{\mathcal{B}}\}$ and the set of final states $F_{\mathcal{B}}$ is reduced to $\{q_f^{\mathcal{B}}\}$. Intuitvely, $\mathcal{B}$ is a synchronized product of $\mathcal{A}$ together with the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$. An extra componant is needed to keep track of the bracketing depth.
The transition function is defined by:

$$
\begin{aligned}
&\delta_{\mathcal{B}}(q_0^{\mathcal{B}}, [) = (\delta_{\mathcal{A}}(s_{\text{in}}, \pi(\bot)), q_0^1, \ldots, q_0^n, 1)\\
&\delta_{\mathcal{B}}((s, q_1, \ldots, q_n, \ell), \sigma) = (s, \delta_1(q_1, \sigma), \ldots, \delta_n(q_n, \sigma), \ell)\\
&\delta_{\mathcal{B}}((s, q_1, \ldots, q_n, \ell), [) = (s, \delta_1(q_1, [), \ldots, \delta_n(q_n, [), \ell + 1)\\
&\delta_{\mathcal{B}}((s, q_1, \ldots, q_n, \ell'),]) = (s, \delta_1(q_1,]), \ldots, \delta_n(q_n,]), \ell' - 1)\\
&\delta_{\mathcal{B}}((s, q_1, \ldots, q_n, 2),]) = (\delta(s, \pi'), q_1^0, \ldots, q_n^0, 1)\\
&\delta_{\mathcal{B}}((s, q_1, \ldots, q_n, 1),]) = (s, q_1^0, \ldots, q_n^0, 0)\\
&\delta_{\mathcal{B}}((s, q_1, \ldots, q_n, 0), q) = q_f \text{ for } q \in Acc(s)
\end{aligned}
$$

for all $\ell \in [2, k+1]$, $\ell' \in [3, k+1]$ and $\sigma \in \Sigma$ and where $\pi(\bot)$ is the boolean vector $(b_1, \ldots, b_n)$ with for $i \in [1, n]$, b_i is equal to 1 if $\bot \in \mathcal{O}_i$ and $b_i = 0$ otherwise and similarly π' is the boolean vector $(b_1, \ldots, b_n)$ where for $i \in [1, n]$, b_i is equal to 1 if $\delta(q_i,]) \in F_i$ and 0 otherwise.

Conversely, let $\mathcal{A} = (Q_{\mathcal{A}}, \delta_{\mathcal{A}}, q_0^{\mathcal{A}}, F_{\mathcal{A}})$ be a deterministic automaton over the alphabet $\Sigma \cup Q \cup \{[,]\}$ accepting words of the form wq with w being a well-bracketed word of depth $k+1$ over $\Sigma \cup \{[,]\}$ and $q \in Q$. We can assume without lost of generality that the set of states of $\mathcal{A}$ is of the form $Q'_{\mathcal{A}} \times [0, k+1]$ and that for all $w \in (\Sigma \cup \{[,]\})^*$, if $\delta_{\mathcal{A}}(q_0^{\mathcal{A}}, w) = (q, \ell)$ then $w]^\ell$ is a well-bracketed word of depth $k+1$.

For each pair (p, q) of states in $Q'_{\mathcal{A}}$, we define the language $\mathcal{O}_{p,q} := \{w \mid \delta((p, 1), w) = (q, 1)\}$ which is a regular set of well-bracketed words of depth k.

We can now define an automaton $\mathcal{B}$ with oracles accepting the configurations of $\mathcal{P}$ corresponding to words accepted by $\mathcal{A}$. Assume that $Q'_{\mathcal{A}}$ is equal to $\{q_1, \ldots, q_m\}$. We pick a bijection ρ from $[1, m]^2$ to $[1, m^2]$.

The automaton $\mathcal{B}$ is formally defined as $(Q_{\mathcal{B}}, Q, \Gamma, \delta_{\mathcal{B}}, q_0^{\mathcal{B}}, \mathcal{O}_0, \ldots, \mathcal{O}_{m^2}, Acc)$.

We take $Q_{\mathcal{B}} := Q'_{\mathcal{A}}$, Γ is the set of all level k stacks over Σ, $q_0^{\mathcal{B}}$ is such that $q_0^{\mathcal{A}} = (q_0^{\mathcal{B}}, 0)$, $\mathcal{O}_0 = \{\bot\}$ and for all $(i, j) \in [1, m]^2$, $\mathcal{O}_{\rho(i,j)} = \mathcal{O}_{q_i, q_j}$.

The transition function of $\delta_{\mathcal{B}}$ is defined for all boolean vector $\pi = (b_0, \ldots, b_{m^2})$ by:

$$\begin{cases} \delta_{\mathcal{B}}(q_0^{\mathcal{B}}, \pi) = p & \text{for } b_0 = 1 \text{ and } \delta_{\mathcal{A}}((q_0^{\mathcal{B}}, 0), [) = (p, 1) \\ \delta_{\mathcal{B}}(q_i, \pi) = q_j & \text{for } b_{\rho(i,j)} = 1 \end{cases}$$

finally, we take $Acc(p) = \{q \in Q \mid \delta_{\mathcal{A}}((p,1),]q) \in F_{\mathcal{A}}\}$ for all $p \in Q'_{\mathcal{A}}$. □

C Proof of Proposition 3

Proposition 3. *Let $\sigma \in (\Gamma \backslash \{\bot\})^*$, $q \in Q$ and $\gamma \in \Gamma \backslash \{\bot\}$. Then Éloïse has a winning strategy in $\mathbb{G}$ from $(q, \bot\sigma\gamma)$ if and only if there exists some $R \in \mathcal{R}(q, \gamma)$ such that $(r, \bot\sigma)$ is winning for Éloïse in $\mathbb{G}$ for every $r \in R$.*

Proof. Assume Éloïse has a winning strategy from $(q, \bot\sigma\gamma)$ in $\mathbb{G}$ and call it φ. Consider the set $\mathcal{L}$ of all plays in $\mathbb{G}$ that starts from $(q, \bot\sigma\gamma)$ and where Éloïse respects φ. Define R to be the (possibly empty) set that consists of all $r \in Q$ such that there is a play in $\mathcal{L}$ of the form $v_0 \cdots v_k(r, \bot\sigma)v_{k+1} \cdots$ where each v_i for $0 \leq i \leq k$ is of the form $(p_i, \bot\sigma\sigma'_i)$ for some $\sigma'_i \neq \varepsilon$. In other words, R consists of all states that can be reached on popping (possibly a rewriting of) γ for the first time in a play where Éloïse respects φ. Define a (partial) function τ as $\tau((q, \bot\sigma\sigma')) = (q, \bot\sigma')$ for every $q \in Q$ and set $\tau^{-1}((q, \bot\sigma')) = (q, \bot\sigma\sigma')$. Then τ^{-1} is extended as a morphism over V^*. It is easily shown that $R \in \mathcal{R}(q, \gamma)$. Indeed a winning strategy for Éloïse in $\mathbb{G}(R)$ is defined as follows:

- if some empty stack configuration has already been visited play any move,
- otherwise goes to $\tau(\varphi(\tau^{-1}(\lambda))$, where λ is the partial play seen so far.

By definition of $\mathcal{L}$ and R it easily follows that the previous strategy is winning for Éloïse in $\mathbb{G}(R)$, and therefore $R \in \mathcal{R}(p, \gamma)$.
Finally, for every $r \in R$ there is, by definition of $\mathcal{L}$ a partial play λ_r that starts from $(q, \bot\sigma\gamma)$, where Éloïse respects φ and that ends in $(r, \bot\sigma)$. A winning strategy for Éloïse in $\mathbb{G}$ from $(r, \bot\sigma)$ is given by $\psi(\lambda) = \varphi(\lambda'_r \cdot \lambda)$, where λ'_r denotes the partial play obtained from λ_r by removing its last vertex $(r, \bot\sigma)$.

Conversely, let us assume that there is some $R \in \mathcal{R}(q, \gamma)$ such that $(r, \bot\sigma)$ is winning for Éloïse in $\mathbb{G}$ for every $r \in R$. For every $r \in R$, let us denote by φ_r a winning strategy for Éloïse from $(r, \bot\sigma)$ in $\mathbb{G}$. Let φ_R be a winning strategy for Éloïse in $\mathbb{G}(R)$ from $(q, \bot\gamma)$. Let us define τ and τ^{-1} as in the direct implication and extend them as (partial) morphism over V^*. Define the following strategy for Éloïse in $\mathbb{G}$ for plays starting from $(q, \bot\sigma\gamma)$. For any partial play λ,

- if λ does not contain a configuration of the form $(p, \bot\sigma)$ then $\varphi(\lambda) = \tau^{-1}(\varphi_R(\tau(\lambda)))$;
- otherwise let $\lambda = \lambda' \cdot (r, \bot\sigma) \cdot \lambda''$ where λ' does not contain any configuration of the form $(p, \bot\sigma)$. From how φ is defined in the previous case, it is follows that $r \in R$. One finally sets $\varphi(\lambda) = \varphi_r(\lambda'' \cdot (r, \bot\sigma))$.

It is then easy to prove that φ is a winning strategy for Éloïse in $\mathbb{G}$ from $(q, \bot\sigma\gamma)$. □

D Proof of Theorem 2

Theorem 2 *The following holds.*

1. *A configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}$ if and only if $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \rho(p_{in}))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
2. *For every $q \in Q$, $\gamma \in \Gamma$ and $R \subseteq Q$, $R \in \mathcal{R}(q, \gamma)$ if and only if $(q, \gamma, (R, \ldots, R), \rho(q))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*

D.1 The game graph $\widetilde{\mathcal{G}}$

Let us first precisely describe the game graph $\widetilde{\mathcal{G}}$. We refer the reader to Figure 1.

- The main vertices of $\widetilde{\mathcal{G}}$ are those of the form $(p, \alpha, \overrightarrow{R}, \theta)$, where $p \in Q$, $\alpha \in \Gamma$, $\overrightarrow{R} = (R_0, \ldots, R_d) \in \mathcal{P}(Q)^{d+1}$ and $\theta \in \{0, \ldots, d\}$. A vertex $(p, \alpha, \overrightarrow{R}, \theta)$ is reached when simulating a partial play Λ in $\mathbb{G}$ such that:
  - The last vertex in Λ is $(p, \bot\sigma\alpha)$ for some $\sigma \in \Gamma^*$.
  - Éloïse claims that she has a strategy to continue Λ in such a way that if α (or a rewriting of it) is eventually popped, the control state reached after popping belongs to R_m, where m is the smallest colour visited since the stack height was at least $|\bot\sigma\alpha|$.
  - The colour θ is the smallest one since the current stack level was reached from a lower stack level.

  A vertex $(p, \alpha, \overrightarrow{R}, \theta)$ is controlled by Éloïse if and only if $p \in Q_{\mathbf{E}}$.
- The vertices *tt* and *ff* are here to ensure that the vectors $\overrightarrow{R}$ encoded in the main vertices are correct. Vertex *tt* is controlled by Abelard, whereas vertex *ff* belongs to Éloïse. As these vertices are dead-ends, a play reaching *tt* is won by Éloïse whereas a play reaching *ff* is won by Abelard (recall that the player controlling the dead-end loses).

There is a transition from some vertex $(p, \alpha, \overrightarrow{R}, \theta)$ to tt, if and only if there exists a transition rule $(r, pop) \in \Delta(p, \alpha)$, such that $r \in R_\theta$ (this means that $\overrightarrow{R}$ is correct with respect to this transition rule). Dually, there is a transition from a vertex $(p, \alpha, \overrightarrow{R}, \theta)$ to ff if and only if there exists a transition rule $(r, pop) \in \Delta(p, \alpha)$ such that $r \notin R_\theta$ (this means that $\overrightarrow{R}$ is not correct with respect to this transition rule).

- To simulate a transition rule $(q, rew(\beta)) \in \Delta(p, \alpha)$, the player that controls $(p, \alpha, \overrightarrow{R}, \theta)$ moves to $(q, \beta, \overrightarrow{R}, \min(\theta, \rho(q)))$. Note that the last component has to be updated as the smallest colour seen since the current stack level was reached is now $\min(\theta, \rho(q))$.

- To simulate a transition rule $(q, push(\beta)) \in \Delta(p, \alpha)$, the player that controls $(p, \alpha, \overrightarrow{R}, \theta)$ moves to $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$. This vertex is controlled by Éloïse who has to give a vector $\overrightarrow{S} = (S_0, \dots, S_d) \in \mathcal{P}(Q)^{d+1}$ that describes the control states that can be reached if β (or a symbol that rewrites it later by applying rew rules) is eventually popped. To describe this vector, she goes to the corresponding vertex $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$.

  Any vertex $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$ is controlled by Abelard who chooses either to simulate a bump or a stair. In the first case, he additionally has to pick the minimal colour of the bump. To simulate a bump with minimal colour i, he goes to a vertex $(s, \alpha, \overrightarrow{R}, \min(\theta, i, \rho(s)))$, for some $s \in S_i$, through an edge coloured by i.

  To simulate a stair, Abelard goes to the vertex $(q, \beta, \overrightarrow{S}, \rho(q))$.

  The last component of the vertex (that stores the smallest colour seen since the currently simulated stack level was reached) has to be updated in all those cases. After simulating a bump of minimal colour i, the minimal colour is $\min(\theta, i, \rho(s))$. After simulating a stair, this colour has to be initialized (since a new stack level is simulated). Its value, is therefore $\rho(q)$, which is the unique colour since the (new) stack level was reached.

The only vertices that are coloured are those of the form $(p, \alpha, \overrightarrow{R}, \theta)$ and the colour of such a vertex is $\rho(p)$. Some edges are also coloured. See Figure 1 for details.

Remark 9. *In the definition of parity games we were required a total colouring function working only on vertices. One can add extra intermediate states to remove colouring on edges and assign the largest colour d to all uncoloured vertices.*

D.2 Proof of Theorem 2

This subsection is devoted to the proof of Theorem 2. We only prove the first part of it, the proof of the second one being a subpart of it.

Factorisation of a play in $\mathbb{G}$.

Recall that for an infinite play $\Lambda = v_0 v_1 \cdots$ in $\mathbb{G}$ $Steps_\Lambda$ denote the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally, $Steps_\Lambda = \{i \in \mathbb{N} \mid \forall j \geq i \ sh(v_j) \geq sh(v_i)\}$, where $sh((q, \bot\gamma_1 \cdots \gamma_n)) = n + 1$. Note that $Steps_\Lambda$ is always infinite and hence induces a factorisation of the play Λ into finite pieces.

Indeed, for any play Λ with $Steps_\Lambda = \{n_0 < n_1 < \cdots\}$, one can define the sequence $(\Lambda_i)_{i \geq 0}$ by setting $\Lambda_i = v_{n_i} \cdots v_{n_{i+1}}$. Note that each of the Λ_i is either a bump or a stair. In the later we designate $(\Lambda_i)_{i \geq 0}$ as the *rounds factorisation* of Λ.

Factorisation of a play in $\widetilde{\mathbb{G}}$.

Recall that in $\widetilde{\mathcal{G}}$ some edges are coloured. Hence, to represent a play, we have to encode this information on edge colouring. We only need to encode the colours in $\{0, \dots, d\}$ that appears when simulating a bump: a play will be represented as a sequence of vertices together with colours in $\{0, \dots, d\}$ that correspond to colours appearing on edges.

For any play in $\widetilde{\mathbb{G}}$, a *round* is a factor between two visits through vertices of the form $(p, \alpha, \overrightarrow{R}, \theta)$. We have the following possible forms for a round:

- The round is of the form $(p, \alpha, \overrightarrow{R}, \theta)(q, \beta, \overrightarrow{R}, \theta)$ and corresponds therefore to the simulation of a rew rule. We designate it as a *top rewriting bump*.

- The round is of the form $(p, \alpha, \overrightarrow{R}, \theta)(p, \alpha, \overrightarrow{R}, \theta, q, \beta)(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})i$ $(s, \alpha, \overrightarrow{R}, \min(\theta, i, \rho(s)))$ and corresponds therefore to the simulation of a rule pushing β followed by a sequence of moves that ends by popping β (or a rewriting of it). Moreover i is the smallest colour encountered while β (or other top stack symbol obtained by applying rew rules successively) was on the stack.

- The round is of the form $(p, \alpha, \overrightarrow{R}, \theta)(p, \alpha, \overrightarrow{R}, \theta, q, \beta)(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$ $(q, \beta, \overrightarrow{S}, \rho(q))$ and corresponds therefore to the simulation of a rule pushing a symbol β leading to a new stack level below which the play will never go. We designate it as a *stair*.

For any play $\lambda = v_0 v_1 v_2 \cdots$ in $\widetilde{\mathbb{G}}$, we consider the subset of indices corresponding to vertices of the form $(p, \alpha, \overrightarrow{R}, \theta)$. More precisely:

$$Rounds_\lambda = \{n \mid v_n = (p, \alpha, \overrightarrow{R}, \theta),\ p \in Q,\ \alpha \in \Gamma,\ \overrightarrow{R} \in \mathcal{P}(Q)^{d+1},\ 0 \leq \theta \leq d\}$$

Therefore, the set $Rounds_\lambda$ induces a natural factorisation of λ into rounds.

Definition 1 (Rounds factorisation)**.** *For a (possibly partial) play $\lambda = v_0 v_1 v_2 \cdots$, we call* rounds factorisation *of λ, the (possibly finite) sequence $(\lambda_i)_{i \geq 0}$ of rounds defined as follows. Let $Rounds_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$, then for all $0 \leq i < |Rounds_\lambda|$, $\lambda_i = v_{n_i} \cdots v_{n_{i+1}}$.*

Therefore, for every $i \geq 0$, the first vertex in λ_{i+1} equals the last one in λ_i. Moreover, $\lambda = \lambda_1 \odot \lambda_2 \odot \lambda_3 \odot \cdots$, where $\lambda_i \odot \lambda_{i+1}$ denotes the concatenation of λ_i with λ_{i+1} without its first vertex.

Finally, the colour *of a round is the smallest colour in $\{0, \ldots, d\}$ appearing in the round.*

In order to prove both implications of Theorem 2, we build from a winning strategy for Éloïse in one game a winning strategy for her in the other game. The main argument to prove that the new strategy is winning is to prove a correspondence between the factorisations of plays in both games.

Direct implication.

Assume that the configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}$, and let Φ be a corresponding winning strategy for her.

Using Φ, we define a strategy φ for Éloïse in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \rho(p_{in}))$. This strategy stores a partial play in $\mathbb{G}$, that is an element in V^* (where V denotes the set of vertices of $\mathcal{G}$). This memory will be denoted Λ. At the beginning Λ is initialized to the vertex $(p_{in}, \bot)$. We first describe φ, and then we explain how Λ is updated. Both the strategy φ and the update of Λ, are described for a round.

Choice of the move. Assume that the play is in some vertex $(p, \alpha, \overrightarrow{R}, \theta)$ for $p \in Q_{\mathbf{E}}$. The move given by φ depends on $\Phi(\Lambda)$:

- If $\Phi(\Lambda) = (r, pop)$, then Éloïse goes to $\#$ (Proposition 5 will prove that this move is always possible).
- If $\Phi(\Lambda) = (q, rew(\beta))$, then Éloïse goes to $(q, \beta, \overrightarrow{R}, \min(\theta, \rho(q)))$.
- If $\Phi(\Lambda) = (q, push(\beta))$, then Éloïse goes to $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$.

In this last case, or in the case where $p \in Q_{\mathbf{A}}$ and Abelard goes to $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$, we also have to explain how Éloïse behaves from $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$. She has to provide a vector $\overrightarrow{S} \in \mathcal{P}(Q)^{d+1}$ that describes which states can be reached if β (or its successors by top rewriting) is popped, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{S}$, Éloïse considers the set of all possible continuations of $\Lambda \cdot (q, \sigma\alpha\beta)$ (where $(p, \sigma\alpha)$ denotes the last vertex of Λ) where she respects her strategy Φ. For each such play, she checks whether some configuration of the form $(s, \sigma\alpha)$ is visited after $\Lambda \cdot (q, \sigma\alpha\beta)$, that is if the stack level of β is eventually left. If it is the case, she considers the first configuration $(s, \sigma\alpha)$ appearing after $\Lambda \cdot (q, \sigma\alpha\beta)$ and the smallest colour i since β and (possibly) its successors by top rewriting were on the stack. For every $i \in \{0, \ldots d\}$, S_i, is exactly the set of states $s \in Q$ such that the preceding case happens. More formally,

$$S_i = \{s \mid \exists\, \Lambda \cdot (q, \sigma\alpha\beta) v_0 \cdots v_k (s, \sigma\alpha) \cdots \text{ play in } \mathbb{G} \text{ where Éloïse respects } \Phi \text{ and s.t. } |v_j| > |\sigma\alpha|,\ \forall j = 0, \ldots, k \text{ and } \min(\{\rho(v_j) \mid j = 0, \ldots, k\} \cup \{\rho(q)\}) = i\}$$

Finally, we set $\overrightarrow{S} = (S_0, \ldots, S_d)$ and Éloïse moves to $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$.

Update of Λ. The memory Λ is updated after each visit to a vertex of the form $(p, \alpha, \overrightarrow{R}, \theta)$. We have three cases depending on the kind of the last round:

- The round is a top rewriting bump and therefore a $(q, rew(\beta))$ action was simulated. Let $(p, \sigma\alpha)$ be the last vertex in Λ, then the updated memory is $\Lambda \cdot (q, \sigma\beta)$.
- The round is a bump, and therefore a bump of colour i (where i is the colour of the round) starting with some action $(q, push(\beta))$ and ending in a state $s \in S_i$ was simulated. Let $(p, \sigma\alpha)$ be the last vertex in Λ. Then the memory becomes Λ extended by $(q, \sigma\alpha\beta)$ followed by a sequence of moves, where Éloïse respects Φ, that ends by popping β and reach $(s, \sigma\alpha)$ while having i as smallest colour. By definition of S_i such a sequence of moves always exists.
- The round is a stair and therefore we have simulated a $(q, push(\beta))$ action. If $(p, \sigma\alpha)$ denotes the last vertex in Λ, then the updated memory is $\Lambda \cdot (q, \sigma\alpha\beta)$.

Therefore, with any partial play λ in $\widetilde{\mathbb{G}}$ in which Éloïse respects her strategy φ, is associated a partial play Λ in $\mathbb{G}$. An immediate induction shows that Éloïse respects Φ in Λ. The same arguments works for an infinite play λ, and the corresponding play Λ is therefore infinite, starts from $(p_{in}, \bot)$ and Éloïse respects Φ in that play. Therefore it is a winning play.

The following proposition is a direct consequence of how φ was defined.

Proposition 5. *Let λ be a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{in}))$, ends in a vertex of the form $(p, \alpha, \overrightarrow{R}, \theta)$, and where Éloïse respects φ. Let Λ be the play associated with λ built by the strategy φ. Then the following holds:*

1. *Λ ends in a vertex of the form $(p, \sigma\alpha)$ for some $\sigma \in \Gamma^*$.*
2. *θ is the smallest visited colour in Λ since α (or a symbol that was later rewritten as α) has been pushed.*
3. *Assume that Λ is extended, that Éloïse keeps respecting Φ and that the next move after $(p, \sigma\alpha)$ is to some vertex (r, σ). Then $r \in R_\theta$.*

Remark 10. *Proposition 5 implies that the strategy φ is well defined when it provides a move to ⌀. Moreover, one can deduce that, if Éloïse respects φ, ff is never reached.*

The preceding remark shows in particular that any finite play ends in some vertex ⌀ and is therefore won by Éloïse. For infinite plays, using the definitions of $\widetilde{\mathcal{G}}$ and φ, we easily deduce the following proposition.

Proposition 6. *Let λ be an infinite play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{in}))$, and where Éloïse respects φ. Let Λ be the associated play built by the strategy φ, and let $(\Lambda_i)_{i\geq 0}$ be its rounds factorisation. Let $(\lambda_i)_{i\geq 0}$ be the rounds factorisation of λ. Then, for every $i \geq 1$ the following hold:*

1. *λ_i is a bump if and only if Λ_i is a bump*
2. *λ_i has colour $mcol_i^\Lambda$.*

Proposition 6 implies that for any infinite play λ in $\widetilde{\mathbb{G}}$ starting from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{in}))$ where Éloïse respects φ, the sequence of visited colours in λ is $(mcol_i^\Lambda)_{i\geq 0}$ for the corresponding play Λ in $\mathbb{G}$. Hence, using Proposition 4 we conclude that λ is winning if and only if Λ is winning. As Λ is winning for Éloïse, it follows that λ is also winning for her.

Converse implication[1]

Assume now that Éloïse has a winning strategy φ in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{\text{in}}))$. Using φ, we build a strategy Φ for Éloïse in $\mathbb{G}$ for plays starting from $(p_{in}, \bot)$.

[1] Note that in order to prove the converse implication one could follow the direct implication and consider the point of view of Abelard. Nevertheless the proof we give here starts from a winning strategy for Éloïse in $\widetilde{\mathbb{G}}$ and deduces a strategy for her in $\mathbb{G}$: this induces a more involved proof but has the advantage to lead to an effective construction of a winning strategy for Éloïse in $\mathbb{G}$ if one has an effective strategy for her in $\widetilde{\mathbb{G}}$

The strategy Φ uses, as memory, an abstract pushdown content Π, to store the complete description of a play in $\widetilde{\mathbb{G}}$. Recall here that a play in $\widetilde{\mathbb{G}}$ is represented as a sequence of vertices together with colours in $\{0, \dots d\}$.

Therefore the abstract pushdown content alphabet of Π is the set of vertices of $\widetilde{\mathcal{G}}$ together with $\{0, \dots, d\}$. In the following, $top(\Pi)$ will denote the top stack symbol of Π while $StCont(\Pi)$ will be the word obtained by reading Π from bottom to top (without considering the bottom-of-stack symbol of Π). In any play where Éloïse respects Φ, $StCont(\Pi)$ will be a play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{\text{in}}))$ and where Éloïse respects her winning strategy φ. Moreover, for any play Λ where Éloïse respects Φ, we will always have that $top(\Pi) = (p, \alpha, \overrightarrow{R}, \theta)$ if and only if the current configuration in Λ is of the form $(p, \sigma\alpha)$. Finally, if Éloïse keeps respecting Φ, and if α (or a symbol that rewrite it later) is eventually popped the configuration reached will be of the form (r, σ) for some $r \in R_i$, where i is the smallest visited colour since α (or some symbol that was later rewritten as α) was on the stack. Initially, Π only contains $(p_{\text{in}}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{\text{in}}))$.

In order to describe Φ, we assume that we are in some configuration $(p, \sigma\alpha)$ and that $top(\Pi) = (p, \alpha, \overrightarrow{R}, \theta)$. We first describe how Éloïse plays if $p \in Q_{\mathbf{E}}$, and then we explain how the stack is updated.

- **Choice of the move.** Assume that $p \in Q_{\mathbf{E}}$ and that Éloïse has to play from some vertex $(p, \sigma\alpha)$. For this, she considers the value of φ on $StCont(\Pi)$.

  If it is a move to ⌀, Éloïse plays an action (r, pop) for some state $r \in R_\theta$. Lemma 1 will prove that such an r always exists.

  If the move given by φ is to go to some vertex $(q, \beta, \overrightarrow{R}, \min(\theta, \rho(q)))$, Éloïse applies the transition $(q, rew(\beta))$.

  If the move given by φ is to go to some vertex $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$, then Éloïse applies the transition $(q, push(\beta))$.

- **Update of Π.** Assume that the last move, played by Éloïse or Abelard, was to go from $(p, \sigma\alpha)$ to some configuration (r, σ). The update of Π is illustrated by figure 2 and explained in what follows. Éloïse pops in Π until she finds some configuration of the form $(p', \alpha', \overrightarrow{R'}, \theta', p'', \alpha'', \overrightarrow{R})$ that is not preceded by a colour in $\{0, \dots, d\}$. This configuration is therefore in the stair that simulates the pushing of α'' onto the stack (here if $\alpha'' \neq \alpha$ then α'' was later rewritten as α). Éloïse updates Π by pushing θ in Π followed by $(r, \alpha', \overrightarrow{R'}, \min(\theta', \theta, \rho(r)))$.

  Assume that the last move, played by Éloïse or Abelard, was to go from $(p, \sigma\alpha)$ to some configu-

ration $(q, \sigma\alpha')$. Then Éloïse update Π by pushing $(q, \alpha', \overrightarrow{R}, \min(\theta, \rho(q)))$.

Assume that the last move, played by Éloïse or Abelard, was to go from $(p, \sigma\alpha)$ to some configuration $(q, \sigma\alpha\beta)$, let $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S}) = \varphi(StCont(\Pi) \cdot (p, \alpha, \overrightarrow{R}, \theta, q, \beta))$. Intuitively, $\overrightarrow{S}$ describes which states Éloïse can force play to reach if β is eventually popped. Éloïse updates Π by successively pushing $(p, \alpha, \overrightarrow{R}, \theta, q, \beta)$, $(p, \alpha, \overrightarrow{R}, \theta, q, \beta, \overrightarrow{S})$, and $(q, \beta, \overrightarrow{S}, \rho(q))$.

The following lemma gives the meaning of the information stored in Π.

Lemma 1. *Let Λ be a partial play in $\mathbb{G}$, where Éloïse respects Φ, that starts from $(p_{in}, \bot)$ and that ends in a configuration $(p, \sigma\alpha)$. We have the following facts:*

1. *$top(\Pi) = (p, \alpha, \overrightarrow{R}, \theta)$ with $\overrightarrow{R} \in \mathcal{P}(Q)^{d+1}$ and $0 \leq \theta \leq d$.*
2. *$StCont(\Pi)$ is a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{in}))$, that ends with $(p, \alpha, \overrightarrow{R}, \theta)$ and where Éloïse respects φ.*
3. *θ is the smallest colour visited since α (or some symbol that was later rewritten as α) was pushed.*
4. *If Λ is extended by some move that pops α, the configuration (r, σ) that is reached is such that $r \in R_\theta$.*

Proof. The proof goes by induction on Λ. We first show that the last point is a consequence of the second and third points. To aid readability, one can refer to Figure 2. Assume that the next move after $(p, \sigma\alpha)$ is to apply an action $(r, pop) \in \Delta(p, \alpha)$. The second point implies that $(p, \alpha, \overrightarrow{R}, \theta)$ is winning for Éloïse in $\widetilde{\mathbb{G}}$. If $p \in Q_{\mathbf{E}}$, by definition of Φ, there is some edge from that vertex to $\#$, which means that $r \in R_\theta$ and allows us to conclude. If $p \in Q_{\mathbf{A}}$, note that there is no edge from $(p, \alpha, \overrightarrow{R}, \theta)$ (winning position for Éloïse) to the losing vertex *ff*. Hence we conclude the same way.

Let us now prove the other points. For this, assume that the result is proved for some play Λ, and let Λ' be an extension of Λ. We have two cases, depending on how Λ' extends Λ:

- Λ' is obtained by applying a rule of type rew or $push$. The result is trivial in that case.
- Λ' is obtained by applying a *pop* rule. Let $(p, \sigma\alpha)$ be the last configuration in Λ, and let $\overrightarrow{R}$ be the last vector component in $top(\Pi)$ when in configuration $(p, \sigma\alpha)$. By the induction hypothesis, it follows that $\Lambda' = \Lambda \cdot$

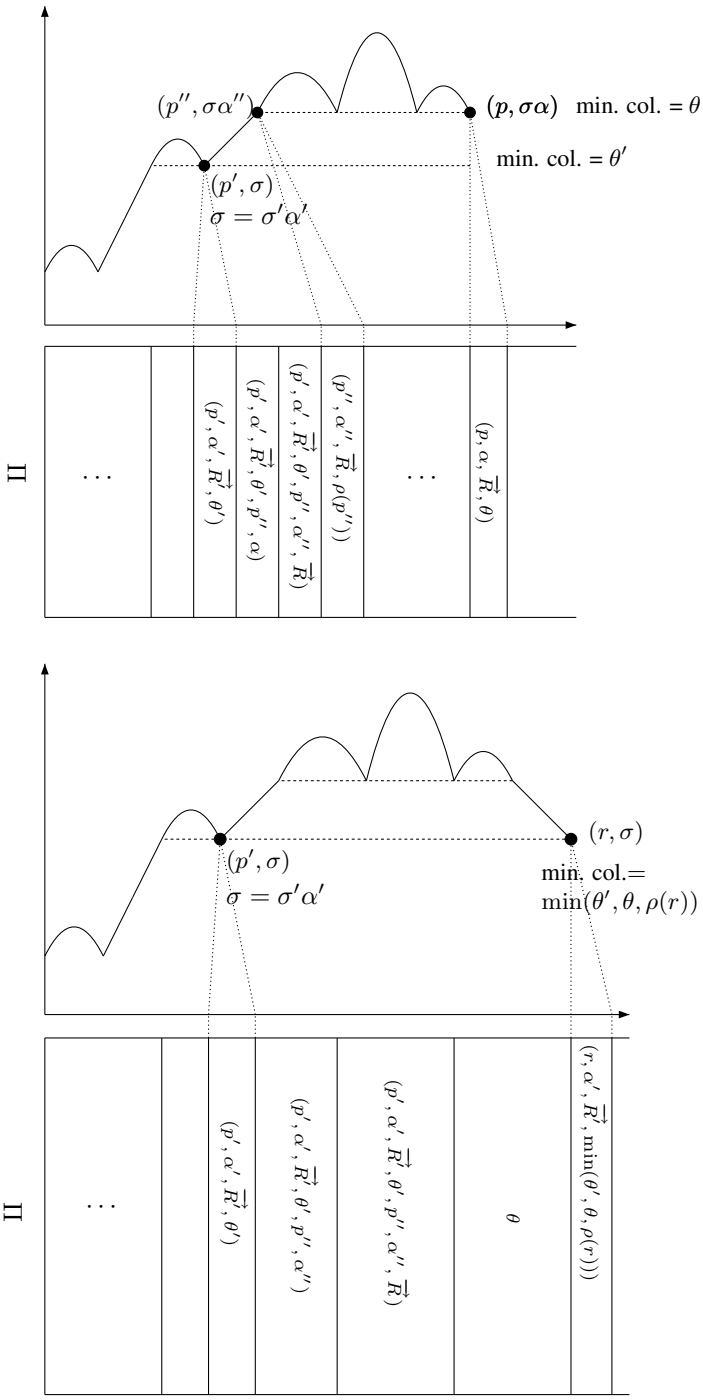

Figure 2. Updating the strategy's stack Π

(r, σ) with $r \in R_\theta$. Considering how Π is updated, and using the fourth point, we easily deduce that the new strategy stack Π is as desired (one can have a look at Figure 2 for more intuition).

□

Actually, we easily deduce a more precise result.

Lemma 2. *Let Λ be a partial play in $\mathbb{G}$ starting from $(p_{in}, \bot)$ and where Éloïse respects Φ. Let $(\Lambda_i)_{i\geq 0}$ be its rounds factorisation. Let $\lambda = StCont(\Pi)$, where Π denotes the strategy's stack in the last vertex of Λ. Let $(\lambda_i)_{i=0,\dots,k}$ be the rounds factorisation of λ. Then the following holds:*

- *λ_i is a bump if and only if Λ_i is a bump.*
- *λ_i has colour $mcol_i^\Lambda$.*

Both lemmas 1 and 2 are for partial plays. A version for infinite plays would allow us to conclude. Let Λ be an infinite play in $\mathbb{G}$. We define an infinite version of λ by considering the limit of the stack contents $(StCont(\Pi_i))_{i\geq 0}$ where Π_i is the strategy's stack after the first i moves in Λ. See [33] for similar constructions. It is easily seen that such a limit always exists, is infinite and corresponds to a play won by Éloïse in $\widetilde{\mathbb{G}}$. Moreover the results of Lemma 2 apply.

Let Λ be a play in $\mathbb{G}$ with initial vertex $(p_{\text{in}}, \bot)$, and where Éloïse respects Φ, and let λ be the associated infinite play in $\widetilde{\mathbb{G}}$. Therefore λ is won by Éloïse. Using Lemma 2 and Proposition 4, we conclude, as in the direct implication that Λ is winning.

E Proof of Theorem 3

Theorem 3. *Assume Éloïse has a winning strategy φ in $\widetilde{\widetilde{\mathbb{G}}}$ that uses a memory ranging from some set M. Then one can construct an abstract pushdown process $\mathcal{T}$ with output that realises a winning strategy Φ for Éloïse in $\mathbb{G}$. Moreover the abstract pushdown alphabet used by $\mathcal{T}$ is $\widetilde{V} \times M$ and, at any moment in a play where Éloïse respects Φ, the abstract pushdown content of $\mathcal{T}$ has exactly the same height as the one in the current position of the game graph. Finally, if φ is effective the same holds for Φ.*

Proof (sketch). The proof strongly relies on the converse implication of the proof of Theorem 2. Hence we assume that the reader read Section D.2 where the following was proved. Starting from a winning strategy for Éloïse in $\widetilde{\widetilde{\mathbb{G}}}$ from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \rho(p_{in}))$ we built a winning strategy Φ for Éloïse in $\mathbb{G}$ that is winning from $(p_{in}, \bot)$. Moreover Φ uses an abstract pushdown content Π as a memory and the stack alphabet of Φ is the set of vertices in $\widetilde{\mathcal{G}}$ augmented by the set of colours $\{0, \dots, d\}$. Assume now that φ is a strategy with memory ranging from some set M. Then, it is fairly easy to note that Π can be modified (and denoted Π') so that it is now over a larger alphabet (namely the Cartesian product of the previous one with M) and that its M component is exactly the value of the memory over $StCont(\Pi)$. Hence the move given by φ over $StCont(\Pi')$ only depends on $top(\Pi')$.

At that point we have the desired result except that Π' is not synchronised (*i.e.* always has the same height) with the abstract pushdown content of the game graph. Indeed Π' stores for instance all bumps on a given level which can arbitrarily increases its height. Nevertheless we can note that the only case where we really use the fact that Π' is a stack is when we simulate a *pop* action and have to remove a lot of symbols from Π' (until finding the step that reached the currently left stack level). When doing this we always go back to the last moment where some stack level was visited. Hence instead of collecting information on bumps in Π' we can forget about it and only recall the information on the last configuration of each stack level. Therefore, if we denote by Π'' this new stack, it contains exactly one element per stack level in the game and therefore always has the same height than the stack in the game. Hence we have the desired result.

So far we have only considered the case where we start from a configuration with an empty stack. In order to deal with the general case, we only have to explain how to initialise Π''. This can be done by following the idea behind Theorem 1. Indeed, consider some (winning) position (p, σ) with $\sigma = \bot\sigma_1 \cdots \sigma_n$. Let $\emptyset \cdot R_0, \cdots R_n$ be the run of the automaton $\mathcal{A}$ from Theorem 1 on $\bot\sigma_1 \cdots \sigma_n$. Then set $v_i = (p, \sigma_i, (R_i, \cdots, R_i), \rho(p))$ and initialise Π'' as $\bot v_0 \cdots v_n$. Now if one defines the winning strategy as in the previous case with this initial value, it leads to a winning strategy from (p, σ). This is immediate if σ_n (and its possibly rewritings) is never popped. Otherwise, it mainly follows from the fact that Éloïse wins from $(r, \bot\sigma_1 \cdots \sigma_{n-1})$ for any $r \in R_n$ (and by induction).

□

F Proof of Theorem 4

Theorem 4. *The sets of winning positions in a higher-order pushdown parity game are regular and can be effectively computed. Computing these regions is an n-*EXPTIME*-complete problem for an order-n pushdown parity game.*

Proof. Regularity of the wining region is directly obtained by combining Property 1 together with Proposition 2 and Theorem 2. Hence, we only discuss the complexity issue. The upper bound comes from two facts:

- The description (by means of a higher-order pushdown automaton) of $\widetilde{\mathbb{G}}$ is exponentially larger than the one of $\mathbb{G}$ (the control state space is exponentially larger).

- Both $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ use the same number of colours

Starting with an order-n parity pushdown game, and applying n time the reduction of Theorem 2, one ends up with a parity game using $(d+1)$ colours over a finite game graph whose size is n times exponential in the size of the original order-n pushdown automaton. As solving this latter game is exponential only on the number of colours [22] the global procedure is in n-EXPTIME.

For the lower bound, it follows from the fact that deciding the winner in an order-n pushdown two-player reachability game is already n-EXPTIME-hard. Indeed, checking emptiness of a nondeterministic higher-order pushdown automaton of order-n is an $(n-1)$-EXPTIME complete problem [17]. Trivially this result is still true if we assume that the input alphabet is reduced to a single letter. Now consider an order-$(n+1)$ nondeterministic higher order pushdown automaton $\mathcal{A}$ whose input alphabet is reduced to a single letter. The language accepted by $\mathcal{A}$ is non-empty if and only if there is a path from the initial configuration of $\mathcal{A}$ to a final configuration of $\mathcal{A}$ in the transition graph G of $\mathcal{A}$. Equivalently the language accepted by $\mathcal{A}$ is non-empty if and only if Éloïse wins the reachability game $\mathbb{G}$ over G where she controls all vertices (and where the play starts from the initial configuration of $\mathcal{A}$ and where final vertices are those corresponding to final configurations of $\mathcal{A}$). Now consider the reduction used to prove Theorem 2 and apply it to $\mathbb{G}$ (and adapt it to the – simpler – reachability winning condition): it leads to an equivalent reachability game $\widetilde{\mathbb{G}}$ that is now played on the transition graph of an order-n pushdown automaton. In the new game graph, the main vertices are of the form (p, s, R): here R is a subset of Q as we no longer need to deal with colours and we no longer need to use the θ component to store information on the smallest colour. The important fact is that R can be forced to be a singleton: this follows from the fact that all vertices in $\mathbb{G}$ are controlled by Éloïse (and from the proof's details). Therefore, one concludes that the size of the game graph associated with $\widetilde{\mathbb{G}}$ is polynomial in the size of $\mathcal{A}$. Hence, one has shown the following: checking emptiness for an order-$(n+1)$ nondeterministic higher order pushdown automaton whose input alphabet is reduced to a single letter can be polynomially reduced to solve a reachability game over the transition graph of an order-n higher order pushdown automaton. In conclusion, this last problem is n-EXPTIME hard (and actually n-EXPTIME-complete). □

G Proof of Theorem 5

Theorem 5. *Consider an order-k parity pushdown game. Then one can construct, for any player, a deterministic order-k pushdown automaton that realises a winning strategy and whose stack has always the same shape as the one in the game.*

Proof (sketch). The order 1 case was noted in Remark 3. For the general case, we reason by induction using Theorem 3 together with the base case (order 1).

Assume the result is proved at order k and consider some order-$(k+1)$ game $\mathbb{G}$. Then using Property 1 we get that $\widetilde{\mathbb{G}}$ is a game of order k. Hence we conclude on the existence of a strategy realised by a deterministic order-k pushdown automaton whose stack has always the same shape as the one in $\widetilde{\mathbb{G}}$. Now apply Theorem 3: it leads to a strategy in $\mathbb{G}$ realised by an abstract pushdown process whose abstract pushdown content is the product of the vertices in $\widetilde{\mathbb{G}}$ together with the memory used to win in $\widetilde{\mathbb{G}}$, that is the product of two order-k stacks having the *same* shape. Hence, this product can be though as a single order-k stack, and therefore the resulting strategy in $\mathbb{G}$ is realised by an order-$(k+1)$ automaton. Finally, the fact that this latter automaton always has the same shape as the current configuration in the game, easily follows by construction (and induction). □

H Proof of Theorem 6

Theorem 6. *The μ-calculus definable sets over transition graphs of higher-order pushdown automata are regular sets of configurations.*

Proof (sketch). Given some μ-calculus formula φ a classical construction [16] leads a finite game graph $\mathcal{G}_\varphi$ whose vertices are all possible subformulas of φ together with a colouring function ρ. Then for any structure $\mathcal{K}$ and any vertex v in $\mathcal{K}$, the following holds: $\mathcal{K}, v \models \varphi$ iff Éloïse wins from (v, φ) in the parity game induced by $\mathcal{K} \times \mathcal{G}_\varphi$. The underlying graph of this game is the cartesian product of $\mathcal{K}$ and $\mathcal{G}_\varphi$, the partition of its vertices as well as the associated colouring function being inherited from the ones for $\mathcal{G}_\varphi$. Then, it follows that $||\mathcal{K}||_\varphi = \{v \mid (v, \varphi) \in W_{\mathbf{E}}\}$ where $W_{\mathbf{E}}$ denotes the winning region for Éloïse in the previous game.

The previous construction does not rely on $\mathcal{K}$ being finite and can be used for instance to solve the μ-calculus model-checking problem against pushdown processes [38]. In the case where $\mathcal{K}$ is the transition graph of some higher-order pushdown process, the game graph $\mathcal{K} \times \mathcal{G}_\varphi$ is also the transition graph of some higher-order pushdown process and the partition of its vertices as well as its associated colouring function only depend on the control states. Hence, the solution of the global μ-calculus model-checking problem for

such a graph can be deduced from the (regular) winning region of a higher-order parity pushdown game by a simple operation that preserves regularity. Hence μ-calculus definable sets are regular.

□