Positional Strategies for Higher-Order Pushdown Parity Games

Arnaud Carayol[1] and Michaela Slaats[2]

[1] IGM–LabInfo, Université Paris-Est & CNRS
arnaud.carayol@univ-mlv.fr
[2] RWTH Aachen, Informatik 7, 52056 Aachen, Germany
slaats@automata.rwth-aachen.de

Abstract. Higher-order pushdown systems generalize pushdown systems by using higher-order stacks, which are nested stacks of stacks. In this article, we consider parity games defined by higher-order pushdown systems and provide a k-Exptime algorithm to compute finite representations of positional winning strategies for both players for games defined by level-k higher-order pushdown automata. Our result is based on automata theoretic techniques exploiting the tree structure corresponding to higher-order stacks and their associated operations.

1 Introduction

Two player games of infinite duration over possibly infinite game graphs (also called arenas) play an important role in computer science and in particular in the domain of automatic verification of infinite-state systems (see [14,19] for surveys). In such games, the vertices of the game graph are partitioned into two sets, one for each player. A play consists in moving a token following the edges of the game graph. The player owning the vertex where the token lies, moves the token. If at some point a player cannot move the token he loses, otherwise the play is infinite. The winning condition of the game describes the set of winning plays for one of the players.

We consider the parity winning condition which plays an important role in the context of verification. In a parity game, each vertex is assigned an integer from a finite range and the winning condition is based on the parity of the smallest integer appearing infinitely often during the play. These games are determined (i.e. from any node, one of the players has a winning strategy) and can be won using positional strategies (i.e. strategies that only depend on the current vertex and not on the whole history of the play) [20]. For these games to be accessible to automatic treatment, we assume that the arena, though infinite, has a finite representation. In this article, the arenas will be given by transition graphs of an extension of pushdown automata. The main algorithmic problems, given the finite description of such an arena, are to determine who is winning from a given vertex and to give finite descriptions of the winning regions as well as of the winning strategies for each player. In the context of automatic verification,

E. Ochmański and J. Tyszkiewicz (Eds.): MFCS 2008, LNCS 5162, pp. 217–228, 2008.

these problems correspond respectively to deciding if the behavior of a system satisfies a property expressed in modal μ-calculus, to giving a finite description of the set of states of the system satisfying the property and to synthesizing a controller for the system against a modal μ-calculus formula [16].

The first class of infinite arenas for which parity games have been studied are the ones defined by pushdown automata. In [18], Walukiewicz gives an EXPTIME algorithm to compute the winner from a given configuration as well as a finite description of a winning strategy for one player. In [3,12] the winning region is shown to be regular when a configuration (q, w) is represented by the word qw. Furthermore, a finite representation of a positional winning strategy for one player can easily be derived from [17].

In this article, we consider parity games defined by an extention of pushdown automata called higher-order pushdown automata. Whereas an ordinary (*i.e.* level-1) pushdown automaton works with a stack of symbols (*i.e.* a level-1 stack), a pushdown automaton of level 2 works with a stack of (level-1) stacks. In addition to pushing a symbol onto and popping a symbol from the top-most level-1 stack, a level-2 pushdown automaton can duplicate or remove the entire top-most (level-1) stack. Pushdown automata of higher levels are defined in a similar way. Recently, the infinite structures defined by these automata have received a lot of attention. In [11], 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 [9,8], the ε-closure of their configuration graphs were shown to be exactly those constructible from finite graphs using natural graph transformations (see [15] for a survey).

We consider these infinite structures as arenas for parity games. In [4], Cachat showed that the winner of a parity game defined by a level-k pushdown automaton starting from a given node can be decided in k-EXPTIME. We provide for each player a finite description of the winning region and of a positional winning strategy. These finite descriptions are based on a notion of regularity for higher-order stacks introduced independently in [6] and in [10]. A set of level-k stacks is said to be regular (for operations) if it can be constructed by applying a regular set of sequences of level-k operations to the empty level-k stack. For usual (level-1) stacks, this notion corresponds to the regularity for words. For higher levels it enjoys most of the good properties of the regular sets of words. In particular these sets form a Boolean algebra and are accepted by a natural model of finite automata. The finite description obtained in this article is expressed in terms of this model of finite acceptors. Our construction is based on tree automata techniques introduced in [17] and already used in [4] to solve these games.

The fact that the notion of regularity by operations can be used to describe the winning regions and the positional winning strategies was already known from [6] and [10] respectively. These results are based on the definability in monadic second-order logic which, though effective, only provide a ck-EXPTIME complexity for some constant $c \geq 2$.

Outline. Section 2 introduces the necessary notions. The main theorem is stated in Section 3 with an outline of its proof (developed in Section 4 and 5).

2 Preliminaries

Higher-order pushdown systems. A *level*-1 *stack* over a finite alphabet Γ can be seen as a word of Γ^*. The empty stack (corresponding to ε) is written $[\,]_1$. We write $Stacks_1(\Gamma) := \Gamma^*$ for the set of all level-1 stacks over Γ. A *level-*(k+1) *stack* for $k \geq 1$ is a non-empty sequence of level-k stacks. The empty stack of level $k+1$ (written $[\,]_{k+1}$) is the level-(k+1) stack containing only the empty stack of level k. The set of all level-(k+1) stacks is defined by $Stacks_{k+1}(\Gamma) := (Stacks_k(\Gamma))^+$. A level-($k$+1) stack s corresponding to the sequence $s_1, \ldots, s_n$ of level-k stacks will be written $[s_1, \ldots, s_n]_{k+1}$ and by convention s_n is the top-most level-k stack of s. We write $top_k(s) = s_n$.

We define the following partial functions on higher-order stacks called *operations*. The level-1 operations are for each symbol $x \in \Gamma$ the operations $push_x$ and pop_x which are respectively defined on level-1 stacks by $push_x([s_0, \ldots, s_n]_1) = [s_0, \ldots, s_n, x]_1)$ and $pop_x([s_0, \ldots, s_n, x]_1) = [s_0, \ldots, s_n]_1$.

For each level $k+1 \geq 2$, we consider the level-($k+1$) operations $copy_k$ which copies the top-most level-k stack and its symmetric operation $\overline{copy}_k$ which removes the top-most level-k stack if it is equal to its predecessor. Formally, these operations are respectively defined on level-(k+1) stacks by $copy_k([s_0, \ldots, s_n]_{k+1}) = [s_0, \ldots, s_n, s_n]_{k+1}$ and $\overline{copy}_k([s_0, \ldots, s_n, s_n]_{k+1}) = [s_0, \ldots, s_n]_{k+1}$. In addition, for each level k, we define a level-k operation written $T_{[\,]_k}$ allowing to test emptiness at level k. Formally $T_{[\,]_k}(s)$ is equal to s if $s = [\,]_k$ and is undefined otherwise.

An operation ψ of level k is extended to stacks of level $\ell > k$ using the equation $\psi([s_0, \ldots, s_n]_\ell) = [s_0, \ldots, \psi(s_n)]_\ell$.

The set of *symmetric operations*[1] of level k over Γ is defined inductively by $Ops_1 = \{push_x, pop_x \mid x \in \Gamma\} \cup \{T_{[\,]_1}\}$ and $Ops_{k+1} = Ops_k \cup \{copy_k, \overline{copy}_k, T_{[\,]_{k+1}}\}$. Moreover, we denote by Ops_k^* the monoid for the composition of partial functions generated by Ops_k.

To obtain a symbolic representation of the operations, we associate to each operation a symbol called an *instruction*. At level 1, we define the set of instructions as $\Gamma_1 = \Gamma \cup \overline{\Gamma} \cup \{\bot_1\}$ where $\overline{\Gamma}$ is a set disjoint from Γ but in bijection with Γ and at level $k+1$, we take $\Gamma_{k+1} = \Gamma_k \cup \{k, \overline{k}, \bot_{k+1}\}$. Furthermore, we define $\Gamma_k^T = \{\bot_\ell \mid \ell \in [1, k]\}$ and $\Gamma_k^O = \Gamma_k \setminus \Gamma_k^T$. We extend the bar notation to all symbols in Γ_k^O by taking $\overline{\overline{x}} = x$. Consider the mapping φ from Γ_k to Ops_k defined by $x \to push_x$, $\overline{x} \to pop_x$, $k \to copy_k$, $\overline{k} \to \overline{copy}_k$ and $\bot_k \to T_{[\,]_k}$ for all $x \in \Gamma, k \in \mathbb{N}$. The mapping φ induces a monoid morphism from Γ_k^* to Ops_k^*. In the following, we will not distinguish between the two monoids and omit φ.

Definition 1. *A* higher-order pushdown system P of level k *(*k*-HOPDS for short) is defined as a tuple* $(Q, \Sigma, \Gamma, \Delta)$ *where* Q *is the finite set of states,* Σ *is the input alphabet,* Γ *is the stack symbol alphabet and* $\Delta \subseteq Q \times \Sigma \times \Gamma_k \times Q$ *is the transition relation.*

[1] The usual definition of higher-order pushdown automata [11] considers the unconditional destruction of level-k stacks written pop_k. The choice of the symmetric operations and its consequences are discussed in the conclusion.

A configuration is a tuple $(p, s) \in Q \times Stacks_k(\Gamma)$. We write $(p, s) \xrightarrow{\alpha} (q, s')$ if there exists a transition $(p, \alpha, \rho, q) \in \Delta$ such that $s' = \rho(s)$. A k-HOPDS is *deterministic* if for all $\alpha \in \Sigma$ and all configurations c, c' and c'', $c \xrightarrow{\alpha} c'$ and $c \xrightarrow{\alpha} c''$ implies $c' = c''$.

Regular sets of higher-order pushdown stacks. The natural notion of regularity for sets of level-1 stacks is the regularity for words. Indeed the set of reachable stack contents of a pushdown automaton is regular [2].

Starting from level 2, two notions of regularity have been introduced: regularity for words and regularity for (symmetric) operations. We will use the second notion and discuss this choice in the conclusion.

The notion of regularity for words was introduced in [1]. A level-k stack is represented by a well-bracketed word of depth k (e.g. the level-2 stack $[[aa]_1[abb]_1]_2$ is represented by the word $[[aa][aab]]$). A set of level-k stacks is *regular for words* if the set of words representing it is a regular set of words. For example the set of level-2 stacks $\{[[a^n]_1[b^m]_1]_2 \mid n, m \geq 0\}$ is regular for words.

The notion of *regularity for (symmetric) operations* was introduced independently in [6] and [10]. A set of level-k stacks is *regular for operations* if it can be obtained by applying a regular subset of Ops_k^* to the empty level-k stack $[\ ]_k$. Formally, we define the set of all level-k stacks which are *regular for operations* as follows: $OReg_k(\Gamma) = Reg(Ops_k^*(\Gamma))([\ ]_k) = Reg(\Gamma_k^*)([\ ]_k)$ (i.e. $S \in OReg_k(\Gamma)$ if there exists $R \in Reg(\Gamma_k^*)$ and $S = \{\rho([\ ]_k) \mid \rho \in R\}$). At level 1, the notion of regularity for operations coincides with notion of regularity for words. For level $k > 2$, every set regular for words is also regular for operations but the converse does not hold. For instance, the set of level-2 stacks $S = \{[[a^n][a^n]]_2 \mid n \geq 0\}$ is regular for operations as $S = push_a^* copy_1([\ ]_2) = a^*1([\ ]_2)$ but it is not regular for words.

For every level $k \geq 1$, the set $OReg_k(\Gamma)$ is a Boolean algebra. These closure properties are due to the fact that a level-k stack s can be uniquely represented by the smallest sequence of instructions $\rho \in \Gamma_k^*$ such that $s = \rho([\ ]_k)$. This unique sequence, called the *reduced sequence* of s, will be written ρ_s. For instance the reduce sequence of the level-2 stack $[[aab][ab]]_2$ is $aab1\bar{b}\bar{a}b$. Note that the reduced sequence of a level-1 stack is simply the stack itself. For a stack of level $k + 1 \geq 2$, its reduced sequence cannot contain $x\bar{x}$, $\perp_\ell$ nor $\bar{k}$ for any $x \in \Gamma_k^O$ and $\ell \in [1, k+1]$. In fact, the reduced sequence of a level-k stack s is the unique sequence $\rho \in \Gamma_k^*$ such that $s = \rho([\ ]_k)$ which does not contain such factors.

The sets in $OReg_k(\Gamma)$ can be characterized by a model of finite automata tightly linked to the notion of reduced sequences. A reduced automaton of level 1 is simply a finite automaton over Γ. A reduced automaton $\mathcal{A}$ of level $k+1 \geq 2$ is given by a tuple (Q, I, F, Δ) together with a finite set of tests $\mathcal{R} \subset OReg_k(\Gamma)$ where Q is the finite set of states, $I \subseteq Q$ and $F \subseteq Q$ are respectively the set of initial and final states, and Δ is the finite set of transitions of the form $p \xrightarrow{\gamma} q, T$ with $\gamma \in \Gamma_k^O \setminus \{\bar{k}\}$ and $T \subseteq R$. Intuitively, the automaton in state p on a stack s can apply the transition to go to state q on the stack $\gamma(s)$ if the top-most level-k

stack of $\gamma(s)$ belongs to T. Furthermore, we impose that the automaton only follows reduced sequences: if $p \xrightarrow{\gamma} p', T \in \Delta$ and $p' \xrightarrow{\gamma'} p'', T \in \Delta$ then $\gamma' \neq \bar{\gamma}$.

A run of $\mathcal{A}$ is a sequence $(q_0, s_0), \ldots, (q_n, s_n) \in (Q \times Stacks_{k+1}(\Gamma))^+$ where $q_0 \in I$, $s_0 = [\,]_{k+1}$ and for all $i \in [0, n-1]$, there exists a transition $q_i \xrightarrow{\gamma_{i+1}} q_{i+1}, T_{i+1}$ with $s_{i+1} = \gamma_{i+1}(s_i)$ and the top-most level k stack of s_{i+1} belongs to T for all $T \in T_{i+1}$. A run of $\mathcal{A}$ accepts a stack s if $q_n \in F$ and $s = s_n$. Note that in this case, the reduce sequence of s is $\gamma_1 \ldots \gamma_n$.

We will always consider reduced automata whose tests are given by reduced automata of one level below. The size of the automaton is the size of the transition relation together with the sum of the reduced automata accepting the set of tests.

Theorem 1 ([6,10]). *For all $k \geq 1$, the sets of level-k stacks regular for operations are exactly those sets accepted by reduced level-k automata. Moreover $OReg_k(\Gamma)$ forms a Boolean algebra.*

Parity games defined by higher-order pushdown systems. A *parity game* $\mathcal{G}$ played between Player 0 and Player 1 is given by a tuple (V_0, V_1, E), where V_i is the set of nodes of Player i for $i \in \{0, 1\}$ and $E \subseteq (V_0 \cup V_1) \times (V_0 \cup V_1)$ is the edge relation, and $\Omega : (V_0 \cup V_1) \to [1, n]$ is the coloring mapping for some fixed $n \in \mathbb{N}$.

Player 0 and Player 1 play in $\mathcal{G}$ 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, he loses the play. Otherwise, the play is an infinite word $\pi \in (V_0 \cup V_1)^\omega$ and is won by Player 0 if the smallest color that is seen infinitely often in π is even.

A strategy for Player i is a partial function φ_i assigning to a partial play ending in some vertex $v \in V_i$ a vertex v' such that $(v, v') \in E$. Player i *respects a strategy* φ_i during some play $\pi = v_0 v_1 v_2 \cdots$ if $v_{i+1} = \varphi_i(v_0 \cdots v_i)$, for all $i \geq 0$ such that $v_i \in V_i$. A strategy φ_i for Player i is *winning* from some position $v \in V_0 \cup V_1$ if every play starting from v where Player i respects φ_i is won by him. A *positional* strategy for Player i is a strategy that only depends on the last vertex of the partial play (i.e. it is a partial function from V_i to $V_0 \cup V_1$). Finally, a vertex $v \in V_0 \cup V_1$ is *winning* for Player i if he has a winning strategy from v, and the winning region W_i consists of all winning vertices for Player i.

The positional determinacy theorem for parity games [20] states that from every vertex either Player 0 or Player 1 has a positional winning strategy. This assertion can be strengthened by saying that Player i has a *global* positional winning strategy φ_i such that φ_i is winning for Player i from all vertices in $Dom(\varphi_i)$ and $Dom(\varphi_i) = W_i \cap V_i$ (see [13]).

Definition 2. *A* higher-order pushdown parity game $\mathcal{G}$ of level k *is given by a deterministic k-HOPDS $P = (Q, \Sigma, \Gamma, \delta)$ together with a partition of the states $Q_0 \uplus Q_1$ and a coloring mapping $\Omega_P : Q \to \mathbb{N}$ and is the game (V_0, V_1, E, Ω) where: $V_0 = Q_0 \times \mathrm{Stacks}_k(\Gamma)$, $V_1 = Q_1 \times \mathrm{Stacks}_k(\Gamma)$, E is the Σ-labeled transition relation of P and Ω is defined for $(p, s) \in Q \times \mathrm{Stacks}_k(\Gamma)$ by $\Omega(p, s) := \Omega_P(p)$.*

The labels in Σ on the transitions do not play any role in the game but together with the hypothesis of determinism of P, they permit to give a simpler description of positional strategies. In fact a positional strategy φ for Player i can be described by a partial function from V_i to Σ, or equivalently by a family $(F_\alpha)_{\alpha\in\Sigma}$ of subsets of V_i (i.e. $F_\alpha := \{(p,s) \in V_i \mid \varphi((p,s)) = \alpha\}$). We say that a positional strategy defined by a family $(F_\alpha)_{\alpha\in\Sigma}$ is regular if all these sets are regular[2] for operations.

Tree automata models. Let Σ and W be two finite alphabets which are respectively a labeling alphabet and a set of directions. A Σ-labeled W-tree t is a partial function from W^* to Σ such that $Dom(t)$ is a non-empty prefix-closed subset of W^*. An element of $Dom(t)$ is a *node* and ε is called the *root* of t. For $d \in W$, a node $wd \in Dom(t)$ is a *d-son* of $w \in Dom(t)$ and w is the *parent* of wd. Let Ξ be a finite alphabet, a Ξ-labeling of a Σ-labeled W-tree t is a $\Sigma \times \Xi$-labeled W-tree t' such that $Dom(t) = Dom(t')$ and such that for $w \in Dom(t')$, $t'(w) = (t(w), \sigma)$ for some $\sigma \in \Xi$.

We consider two-way alternating parity tree automata which can from a node of an input tree send several copies to sons of this node but also to its parent. To navigate through the tree, we consider an extended set of directions $ext(W) := W \uplus \{\varepsilon, \uparrow\}$. The symbol $\uparrow$ means "go to the parent node" and ε means "stay on the present node". We take $\forall u \in W^*, d \in W, u.\varepsilon = u$ and $ud \uparrow = u$. The node $\varepsilon\uparrow$ is not defined. As we consider non-complete W-trees (i.e. $Dom(t) \neq W^*$), we assume that the labeling of a tree provides the directions to all sons of a node in the tree: the automaton runs on $\mathcal{P}(W) \times \Sigma$-labeled W-trees t where for all $w \in Dom(t)$, $t(w) = (\theta_w, \sigma_w)$ where $\theta_w = \{d \in W \mid wd \in Dom(t)\}$.

Definition 3. *A* two-way alternating parity tree automaton *(2-PTA for short) running over $\mathcal{P}(W) \times \Sigma$-labeled W-trees is a tuple $\mathcal{A} = (Q, \Delta, I, \Omega)$ where Q is the finite set of states, $\Delta \subseteq Q \times (\mathcal{P}(W) \times \Sigma) \times \mathcal{P}(ext(W) \times Q)$ is the transition relation, I is set of initial states and $\Omega : Q \to \mathbb{N}$ the coloring mapping.*

A transition $(q, (\theta, \sigma), \{(d_1, q_1), \ldots, (d_n, q_n)\}) \in \Delta$ will be written $q, (\theta, \sigma) \to (d_1, q_1) \wedge \ldots \wedge (d_n, q_n)$. We will always assume that $\{d_1, \ldots, d_n\}$ is a subset of $\theta \cup \{\uparrow, \varepsilon\}$. The behavior of a 2-PTA $\mathcal{A} = (Q, \Delta, I, \Omega_\mathcal{A})$ over a $\mathcal{P}(W) \times \Sigma$-labeled W-tree t is given by the parity game $G_{\mathcal{A},t} = (V_0, V_1, E)$ played between two players called Automaton and Pathfinder. The set V_0 of vertices of Automaton is $Dom(t) \times Q$ and the set V_1 of vertices of Pathfinder is $Dom(t) \times \Delta$. For all $w \in Dom(t)$ and $q \in Q$, there is an edge $((w,q),(w,\delta)) \in E$ for all transitions $\delta \in \Delta$ of the form $q, t(w) \to P$. Conversely for every transition $\delta = q, t(w) \to P \in \Delta$, there is an edge $((w,\delta),(wd_i,q_i))$ for all $(d_i,q_i) \in P$. The automaton $\mathcal{A}$ accepts the tree t if Automaton wins $G_{\mathcal{A},t}$ from some vertex in $\{\varepsilon\} \times I$.

The classical notion of (one-way) non-deterministic parity tree automata (PTA) coincide with 2-PTAs with transitions of the form $q, (\theta, \sigma) \to (d_1, q_1) \wedge \ldots \wedge (d_n, q_n)$ where for all $i, j \in [1, n]$, $d_i \in W$ and $d_i = d_j$ implies $i = j$.

[2] We represent a configuration (p, s) by the stack $push_p(s)$.

3 Main Theorem and Outline of the Proof

Theorem 2. *Given a pushdown parity game of level k, we can construct in k-*Exptime *reduced level-k automata describing the winning region and a global positional winning strategy for each player.*

In Section 4, we define for every level k, a tree $\mathbf{t}_k$ (see e.g. Fig. 1) associated to the stacks of level k. The branches of $\mathbf{t}_k$ correspond to the reduced sequences of the level-k stacks. Starting with a parity game $\mathcal{G}$ described by a level-k pushdown automaton P, we construct a 2-PTA $\mathcal{A}_P$ running on $\mathbf{t}_k$ which captures the game $\mathcal{G}$ and whose size is polynomial in the size of P. More precisely, we can reduce the computation of regular representations of a global positional winning strategy and of the winning region for each player to the computation of a regular representation of a global positional winning strategy for the automaton $\mathcal{A}_P$ running on $\mathbf{t}_k$. Intuitively such a strategy consists for every node u of the tree and every state q of the automaton to either provide a transition of the automaton starting with state q which can be applied at node u or a set of directions and states which refute any transitions of the automaton that can be applied at node u in state q.

In Section 5, we show how to compute regular global positional winning strategies for a 2-PTA running on the trees $\mathbf{t}_k$. The proof proceeds by induction on the level of the tree. First based on a construction from [17], we construct for any two-way alternating parity tree automaton $\mathcal{A}$ a non-deterministic one-way parity tree automaton $\mathcal{B}$ accepting the labelings of $\mathbf{t}_k$ corresponding to global positional winning strategies of $\mathcal{A}$ (see Proposition 3). Second for any non-deterministic one-way parity tree automaton $\mathcal{B}$ running on $\mathbf{t}_k$, we construct a two-way alternating parity tree automaton $\mathcal{C}$ running on $\mathbf{t}_{k-1}$ (see Proposition 4) such that from a global positional strategy of $\mathcal{C}$ over $\mathbf{t}_{k-1}$ defined by regular sets of level-$(k-1)$ stacks, we can construct a strategy (for $\mathcal{A}$) accepted by $\mathcal{B}$ defined by regular sets of level-k stacks.

4 From Games to Trees

In this section, we introduce the infinite tree $\mathbf{t}_k$ associated to the stacks of level k and based on the reduced sequences of these stacks. We show that the problem of computing a global positional winning strategy of a level-k pushdown parity game can be reduced in polynomial time to the problem of computing a global positional winning strategy for an alternating two-way parity tree automaton running on the tree $\mathbf{t}_k$.

As we have seen in Section 2, a level-k stack s is uniquely characterized by its reduced sequence ρ_s. Hence the set of reduced sequences of all level-k stacks is a Γ_k^O-tree in which each node corresponds to one and only one level-k stack. In order to increase the expressivity of tree automata running on these trees, we label each node by a finite information about the surrounding of the stack corresponding to this node. The surrounding of a stack $s \in \mathrm{Stacks}_k(\Gamma)$ is a triple $\ell(s) = (d, D, e)$ where:

- $d \in \Gamma_k^O \cup \{\diamond\}$ is the last symbol of ρ_s if $\rho_s \neq \varepsilon$ and is equal to $\diamond$ otherwise,
- D is the set $\{\gamma \in \Gamma_k^O \mid \exists s' \in \mathrm{Stacks}_k(\Gamma), \rho_{s'} = \rho_s\gamma\}$,
- $e \in [0,k]$ is the maximum of $\{n \in [1,k] \mid \bot_n(s) = s\} \cup \{0\}$.

Formally, the tree $\mathbf{t}_k$ is defined for all $s \in \mathrm{Stacks}_k(\Gamma)$ by $\mathbf{t}_k(\rho_s) = \ell(s)$. When referring to the nodes of $\mathbf{t}_k$, we do not distinguish between the stack and its reduced sequence. In particular, we said that a Ξ-labeling t of $\mathbf{t}_k$ is regular if for all $x \in \Xi$, the set of level-k stacks $\mathcal{S}_x = \{s \in \mathrm{Stacks}_k(\Gamma) \mid t(\rho_s) = (\mathbf{t}_k(\rho_s), x)\}$ is regular for operations. A finite representation of t is then given by a family of reduced level-k automata $(\mathcal{A}_x)_{x\in\Xi}$. For $\Gamma = \{a,b\}$, the tree $\mathbf{t}_1$ is essentially the full binary tree. The tree $\mathbf{t}_2$ (depicted in Figure 1) is not complete nor regular.

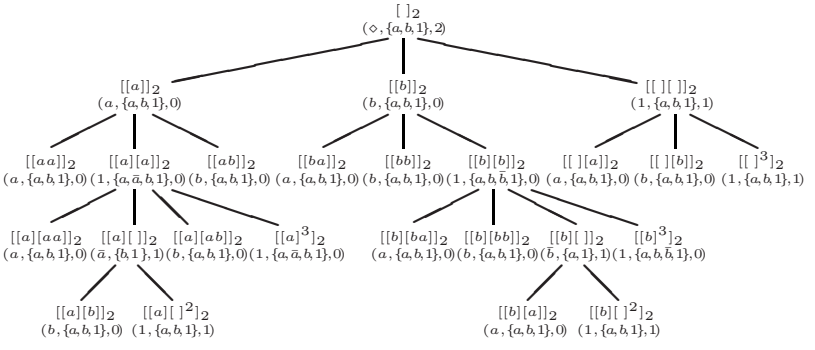

Fig. 1. The tree $\mathbf{t}_2$ for $\Gamma = \{a,b\}$ where the labels appear in parenthesis below the corresponding node

As it was done for (level-1) pushdown parity games in [17], we reduce the decision problem for level-k parity games to the acceptance problem for alternating two-way parity tree automata running over $\mathbf{t}_k$. Intuitively, the non-determinism of the automaton is used to reflect the choices of Player 0 and the alternation is used to reflect the choices of Player 1.

Proposition 1. *Given a pushdown parity game $\mathcal{G}$ of level k, we can construct a 2-PTA $\mathcal{A}$ running on $\mathbf{t}_k$ such that Player 0 wins $\mathcal{G}$ from $(q, [\,]_k)$ if and only if $\mathcal{A}$ has an accepting run on $\mathbf{t}_k$ starting from state q. Furthermore the size of $\mathcal{A}$ is polynomial in the size of the level-k pushdown automaton defining $\mathcal{G}$.*

Proof (Sketch). Let $\mathcal{G}$ be a parity game defined by a level-k pushdown automaton $P = (Q_P, \Sigma, \Gamma, \Delta_P)$ with $Q_P = Q_0 \uplus Q_1$ and a coloring mapping Ω_P. We construct the 2-PTA $\mathcal{A} = (Q_A, \Delta_A, \Omega_A)$ with $Q_A = Q_P$, $\Omega_A = \Omega_P$. To define Δ_A, we introduce for a surrounding $\tau = (d, D, e)$ and an instruction $\gamma \in \Gamma_k$ the direction $[\![\gamma]\!]_\tau$ on $\mathbf{t}_k$. It is defined by $[\![\gamma]\!]_\tau = \gamma$ if $\gamma \in D$, $[\![\gamma]\!]_\tau = \uparrow$ if $\gamma = \overline{d}$, $[\![\gamma]\!]_\tau = \varepsilon$ if $\gamma = \bot_j$ and $e \geq j$ and $[\![\gamma]\!]_\tau$ is undefined otherwise.

For $\mathbf{p} \in \mathbf{Q_0}$, $(p, \alpha, \gamma, q) \in \Delta_P$ and surrounding $\tau = (d, D, e)$ such that $[\![\gamma]\!]_\tau$ is defined, we have: $(p, \tau) \to ([\![\gamma]\!]_\tau, q) \in \Delta_A$.

For $\mathbf{p} \in \mathbf{Q_1}$, let $\delta_1, \ldots, \delta_n$ be the set of all transitions in Δ_P starting in p, i.e. having the form $\delta_i = (p, \alpha_i, \gamma_i, q_i)$ for all $i \in [1, n]$. For all labelings $\tau = (d, D, e)$, we take: $(p, \tau) \to \bigwedge_{i \in [1,n] \wedge [\![\gamma_i]\!]_\tau \text{ def.}} ([\![\gamma_i]\!]_\tau, q_i) \in \Delta_A$.

Note that the size of $\mathcal{A}$ is exponential in the size of Γ_k. □

The relation between the game $\mathcal{G}$ and the 2-PTA $\mathcal{A}$ constructed in Proposition 1 can be lifted to strategies. Before stating this correspondence, we show how to represent a pair of global positional winning strategies φ_{aut} and φ_{path} in $\mathcal{G}(\mathcal{A}, \mathbf{t}_k)$ for Automaton and Pathfinder respectively as a labeling of $\mathbf{t}_k$ by a finite amount of information. The labeling set is $\mathcal{F}_0 \times \mathcal{F}_1$ where $\mathcal{F}_0$ is the set of all partial functions from Q to Δ and $\mathcal{F}_1$ is the set of all partial functions from Q to $\mathcal{P}(ext(W) \times Q)$.

The strategy φ_{aut} of Automaton at a node $w \in Dom(t)$ is given by a partial function ν_0^w from Q to Δ which when defined on a state q gives the transition to apply in the configuration (w, q). Formally, for all $q \in Q$, $\nu_0^w(q) = \delta$ iff $\varphi_{\text{aut}}(w, q) = (w, \delta)$.

The strategy φ_{path} of Pathfinder can be given by a partial function ν_1^w from Q to $\mathcal{P}(\text{ext}(W) \times Q)$. For all $q \in Q$, we have two cases depending on who wins the game from (w, q). If Automaton wins from (w, q) (i.e. there exists a transition $\delta = (q, t(w)) \to P$ such that $\varphi_{\text{path}}(w, \delta)$ is undefined) then $\nu_1^w(q)$ is undefined. If Pathfinder wins from (w, q) then for all transitions $\delta_1, \ldots, \delta_n$ starting with $(q, t(w))$ (i.e. $\delta_j = q, t(w) \to P_j$), we have $\varphi_{\text{path}}(w, \delta_j) = (wd_{j_i}, q_{j_i})$ for some $(d_{j_i}, p_{j_i}) \in P_j$ for all $j \in [1, n]$. In this case, $\nu_1^w(q)$ is equal to $\{(d_{j_i}, q_{j_i}) \mid j \in [1, n]\}$. Intuitively $\nu_1^w(q)$ is defined if Pathfinder wins from (w, q) and in this case it corresponds to a set of directions and states that can refute any transitions of the automaton that can be applied in this configuration. Note that for any node $w \in Dom(t)$, $Dom(\nu_0^w)$ and $Dom(\nu_1^w)$ form a partition of Q.

Conversely we say that a $\mathcal{F}_0 \times \mathcal{F}_1$-labeling of $\mathbf{t}_k$ is a global winning strategy for $\mathcal{A}$ on $\mathbf{t}_k$ if it induces a pair of global winning strategies for each player.

Proposition 2. *Let $\mathcal{G}$ be a pushdown parity game of level k and $\mathcal{A}$ the alternating two-way parity tree automaton given by Proposition 1. Given a regular global positional winning strategy Φ for $\mathcal{A}$ on $\mathbf{t}_k$, we can compute, for each player, a regular global positional winning strategy and a regular representation of the winning region.*

5 Computing Strategies over $\mathbf{t_k}$

In this section, we show how to compute a regular global positional strategy for a 2-PTA $\mathcal{A}$ running on $\mathbf{t}_k$. For this we proceed by induction on the level k.

The first step is based on [17] and consists in showing that for any 2-PTA $\mathcal{A}$ running on $\mathbf{t}_k$, one can construct a PTA $\mathcal{B}$ accepting the $\mathcal{F}_0 \times \mathcal{F}_1$-labelling of $\mathbf{t}_k$ representing a global winning strategy for $\mathcal{A}$ on $\mathbf{t}_k$. In [17], a PTA $\mathcal{B}$ is constructed that accepts the trees representing a positional strategy for Automaton winning from a given state of $\mathcal{A}$ (see [3] for a detailed presentation of the construction). The following proposition simply adapts the construction to make it symmetric between Automaton and Pathfinder.

Proposition 3. *Given a 2-PTA $\mathcal{A}$ running on $\mathbf{t}_k$, we can construct a PTA $\mathcal{B}$ accepting the trees representing global positional winning strategies of $\mathcal{A}$. Furthermore, the size of $\mathcal{B}$ is exponential in the size of $\mathcal{A}$ but the number of colors of $\mathcal{A}$ is linear in the number of colors of $\mathcal{B}$.*

Using Proposition 3, we have reduced our initial problem to computing a regular Ξ-labeling of $\mathbf{t}_{k+1}$ accepted by a given PTA $\mathcal{B}$. In the next step, we reduce this problem to the computation of a global winning strategy for a 2-PTA $\mathcal{C}$ running on $\mathbf{t}_k$.

The construction is based on the fact that $\mathcal{B}$ does not use the $\uparrow$ direction and hence when running on $\mathbf{t}_{k+1}$ only take directions in $\Gamma_k^O \cup \{k\}$. From the point of view of the operations on the stacks, $\mathcal{B}$ does not perform the $\overline{copy}_k$ operation and hence can only access the top-most level-k stack. Intuitively, we can simulate the same behavior at one level below by replacing the direction k by ε: we use alternation instead of performing the $copy_k$ operation. The construction is a bit more technical as we need to relate the surroundings in $\mathbf{t}_{k+1}$ which belong to a node corresponding to a level-$(k+1)$ stack s to the surroundings in $\mathbf{t}_k$ of the node corresponding to the level-k stack $top_k(s)$.

Proposition 4. *Given a PTA $\mathcal{B}$ accepting at least one Ξ-labeling of $\mathbf{t}_{k+1}$, we can construct a 2-PTA $\mathcal{C}$ running on $\mathbf{t}_k$ such that given a regular global positional strategy for $\mathcal{C}$ on $\mathbf{t}_k$, we can construct a regular Ξ-labeling of $\mathbf{t}_{k+1}$ accepted by $\mathcal{B}$. Furthermore, the size of $\mathcal{C}$ is polynomial in the size of $\mathcal{B}$.*

Proof (Sketch). Let $\mathcal{B} = (Q_B, \Delta_B, I_B, \Omega_B)$ be a PTA accepting Ξ-labelings of $\mathbf{t}_{k+1}$. We can assume w.l.o.g that $\mathcal{B}$ runs on $\mathbf{t}_{k+1}$ and guesses the Ξ-labeling (i.e. $Q_B = {Q'}_B \times \Xi$). Furthermore, we can assume that the surroundings appearing in transitions of $\mathcal{B}$ really occur in $\mathbf{t}_{k+1}$.

We define the 2-PTA $\mathcal{C} = (Q_C, \Delta_C, I_C, \Omega_C)$ with $Q_C = Q_B \times \Gamma_k^O \cup \{k, \diamond\}$, $I_C = I_B \times \{\diamond\}$ and $\Omega_C(p, d) = \Omega_B(p)$ for all $d \in \Gamma_k^O \cup \{k, \diamond\}$. For each transition $\delta := q, (d, D, e) \to (\gamma_1, q_1) \wedge \ldots \wedge (\gamma_n, q_n) \in \Delta_B$ and for each $d' \in \Gamma_k^O \cup \{\diamond\}$, we add the following transition to Δ_C:

$$\delta_{\downarrow_{d'}} := (q, d), (d', D', e') \to (\gamma'_1, (q_1, \gamma_1)) \wedge \ldots \wedge (\gamma'_n, (q_n, \gamma_n))$$

when

1. $d = \diamond \Rightarrow d' = \diamond$ and $d' \neq \diamond \wedge d' \neq d \Rightarrow \overline{d'} \in D$ $D' = (D \cup \{\overline{d}\}) \setminus \{k, \overline{d'}, \overline{k}\}$ and $e' = \min\{e, k\}$
2. for all $i \in [1, n]$, γ'_i is equal to ε if $\gamma_i = k$, to $\uparrow$ if $\gamma_i = \overline{d'}$ and to γ_i otherwise.

Intuitively, the automaton $\mathcal{C}$ simulates the actions of $\mathcal{B}$ on the top-most level-k stack. This is enough to capture the whole behavior of $\mathcal{B}$ as $\mathcal{B}$ never performs the $\overline{copy}_k$ operation. Condition 1 relates the surroundings (d, D, e) in $\mathbf{t}_{k+1}$ of a node correspond to a level-$(k+1)$ stack s to the surroundings (d', D', e') in $\mathbf{t}_k$ of the node corresponding to the level-k stack $top_k(s)$. Condition 2 reflects the fact that the $copy_k$ operation does not modify the top-most level-k stack (i.e the direction k is replace by ε).

An important property for the transitions of the 2-PTA $\mathcal{C}$ is that for every $\delta \in \Delta_C$, there exists a unique transition $\delta^{\uparrow} \in \Delta_B$ and a unique $d' \in \Gamma_k^O \cup \{\diamond\}$ such

that $\delta = (\delta^{\uparrow})_{\downarrow_{d'}}$. Moreover, if the labels of δ and $\delta^{\uparrow}$ are respectively (d', D', e') and (d, D, e), we have $D = (D' \cup \{\overline{d'}, k\}) \setminus \{\overline{d}\}$. This property allows us to lift a regular positional strategy for $\mathcal{C}$ on $\mathbf{t}_k$ to a regular positional strategy of $\mathcal{B}$ on $\mathbf{t}_{k+1}$. Consequently, we can compute a regular Ξ-labeling accepted by $\mathcal{B}$ following the regular global positional strategy for $\mathcal{B}$. □

By an induction on the levels combining Proposition 3 and Proposition 4, we reduce the initial problem to the problem of computing a winning strategy in a finite parity game which is k-fold exponential in the size of original automaton. Indeed only the first step provides an exponential blow-up in the number of states but the number of colors remains linear. The computation of a winning strategy on finite parity games is only exponential in the number of colors [20]. Hence, we obtain a k-EXPTIME procedure.

Theorem 3. *Given a 2-PTA $\mathcal{A}$ running on $\mathbf{t}_{k+1}$, we can compute in k-EXPTIME a regular global positional strategy for $\mathcal{A}$ on $\mathbf{t}_{k+1}$.*

6 Conclusion

We have presented a k-EXPTIME algorithm to provide a finite representation of the winning regions and global positional winning strategies in higher-order pushdown parity games of level k. Note that deciding the winner of these games is already k-EXPTIME hard [5]. Our results can be extended to richer graph structures based on higher-order pushdown automata such as for example rooted higher-order pushdown parity games (where the game graph is restricted to the configurations reachable from a given configuration) and their ε-closure (see [8]).

One of the key feature of our approach is the use of the symmetric destruction of level-k stacks $\overline{copy}_k$ instead of the usual unconditional destruction pop_k. This choice is motivated by the closure properties of the notion of regularity induced by the set of symmetric operations as well as the tree structure it induces. The game graphs obtained when considering pop_k instead of $\overline{copy}_k$ can be obtained as ε-closure of rooted higher-order pushdown parity games (see [6]) and hence can be treated in our framework.

In [7], it was shown that when considering higher-order pushdown parity games defined using the unconditional destruction pop_k instead of the symmetric destruction $\overline{copy}_k$ considered in this article, the winning region is regular by words. This result is stronger than the one obtainable by our approach as we can only prove that the winning region is regular for operations. It is important to note that when considering the symmetric version this result no longer holds. The proof of [7] also provides a finite description of a winning strategy from a given vertex for one of the players based on a higher-order pushdown automaton reading the moves of the play and outputting the next move. It is unknown if the notion of regularity by words can be used to describe positional winning strategies for higher-order pushdown parity games.

Acknowledgments. The authors thank Olivier Serre, Wolfgang Thomas and an anonymous referee for their numerous helpfull comments.

References

1. Bouajjani, A., Meyer, A.: Symbolic Reachability Analysis of Higher-Order Context-Free Processes. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 135–147. Springer, Heidelberg (2004)
2. Büchi, J.: Regular canonical systems. Arch. Math. Logik Grundlag. 6, 91–111 (1964)
3. Cachat, T.: Symbolic Strategy Synthesis for Games on Pushdown Graphs. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 704–715. Springer, Heidelberg (2002)
4. Cachat, T.: Higher-order-pushdown automata, the Caucal hierarchy of graphs and parity games. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 556–569. Springer, Heidelberg (2003)
5. Cachat, T., Walukiewicz, I.: The complexity of games on higher order pushdown automata. In: Electronic version (May 2007)
6. Carayol, A.: Regular Sets of Higher-Order Pushdown Stacks. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 168–179. Springer, Heidelberg (2005)
7. Carayol, A., Hague, M., Meyer, A., Ong, C.-H.L., Serre, O.: Winning regions of higher-order pushdown games. In: Proc. LICS 2008 (2008)
8. Carayol, A., Wöhrle, S.: The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 112–123. Springer, Heidelberg (2003)
9. Caucal, D.: On Infinite Terms Having a Decidable Monadic Theory. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420. Springer, Heidelberg (2002)
10. Fratani, S.: Automates á piles de piles.. de piles. PhD thesis, Universit Bordeaux 1 (2005)
11. Knapik, T., Niwiński, D., Urzyczyn, P.: Higher-Order Pushdown Trees Are Easy. In: Nielsen, M., Engberg, U. (eds.) FOSSACS 2002. LNCS, vol. 2303. Springer, Heidelberg (2002)
12. Serre, O.: Note on winning positions on pushdown games with omega-regular conditions. IPL 85(6), 285–291 (2003)
13. Thomas, W.: Languages, automata, and logic. In: Handbook of Formal Languages, vol. 3, pp. 389–455. Springer, Heidelberg (1997)
14. Thomas, W.: Infinite Games and Verification. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, Springer, Heidelberg (2002)
15. Thomas, W.: Constructing Infinite Graphs with a Decidable MSO-Theory. In: Rovan, B., Vojtas, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 113–124. Springer, Heidelberg (2003)
16. Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (2005)
17. Vardi, M.Y.: Reasoning about the Past with Two-Way Automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 628–641. Springer, Heidelberg (1998)
18. Walukiewicz, I.: Pushdown processes: Games and model checking. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 62–74. Springer, Heidelberg (1996)
19. Walukiewicz, I.: A landscape with games in the background. In: Proc. LICS 2004, pp. 356–366. IEE (2004)
20. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. TCS 200, 135–183 (1998)