Visibly Pushdown Games

Christof Löding[1,⋆] P. Madhusudan[2,⋆⋆] Olivier Serre[1,⋆]

[1] LIAFA, Université Paris VII, France
[2] University of Pennsylvania

Abstract. The class of visibly pushdown languages has been recently defined as a subclass of context-free languages with desirable closure properties and tractable decision problems. We study visibly pushdown games, which are games played on visibly pushdown systems where the winning condition is given by a visibly pushdown language. We establish that, unlike pushdown games with pushdown winning conditions, visibly pushdown games are decidable and are 2Exptime-complete. We also show that pushdown games against Ltl specifications and Caret specifications are 3Exptime-complete. Finally, we establish the topological complexity of visibly pushdown languages by showing that they are a subclass of Boolean combinations of $\mathbf{\Sigma_3}$ sets. This leads to an alternative proof that visibly pushdown automata are not determinizable and also shows that visibly pushdown games are determined.

1 Introduction

The theory of two-player games on graphs is a prominent area in formal verification and automata theory. The peculiar acceptance conditions used in the study of automata on infinite words and trees, result in a theory of infinite games that serves as a simple and unified framework for various proofs and constructions in automata theory. In particular, the determinacy theorem for these games and the solvability of infinite games on finite graphs are closely related to the decidability of the monadic second-order logic on trees [14, 16].

In formal verification, infinite games are useful in two contexts. First, the model-checking problem for the μ-calculus is intimately related to solving parity games [6], the precise complexity of which is still open. Second, the theory of games form a natural abstraction of the synthesis and control-synthesis problems, where the aim is to synthesize a system that satisfies a given specification [9].

While most results in model checking involve problems on finite graphs, abstraction of data from software programs with procedures results in pushdown models, where the stack is required to maintain the call-stack of the program. Formal verification of these models against regular specifications is however tractable since emptiness of pushdown automata is decidable. In fact, a variety of

⋆ Supported by the European Community Research Training Network "Games and Automata for Synthesis and Validation" (games).

⋆⋆ Supported partially by ARO URI award DAAD19-01-1-0473, and NSF awards ITR/SY 0121431 and CCR 0306382.

program analysis questions, static code analysis, and compiler optimization can be reduced to reachability in pushdown models [10] and contemporary software model-checking tools such as SLAM [3] implement these decision procedures.

Although checking software models against regular specifications is useful, there are important context-free requirements—specification of pre-post conditions for procedures, security properties that require stack inspection, etc. Recently, a temporal logic called CARET [1] has been defined which allows specification of such context-free properties and yet preserves decidability of pushdown model-checking.

In [2], the class of *visibly pushdown languages* (VPL) is proposed as an automata theoretic generalization of CARET. These languages are accepted by visibly pushdown automata (VPA), which are pushdown automata whose stack-operations are determined by the input. Like the class of regular languages, VPL is closed under all Boolean operations; moreover, decision problems such as inclusion, which are undecidable for context-free languages, are decidable for VPL. VPL includes the class of languages defined by CARET and forms a robust subclass of context-free languages [2].

Turning back to games, pushdown games with parity winning conditions are known to be decidable [15]. This shows that pushdown games with any external ω-regular winning condition can also be solved. However, it is easy to see that solving pushdown games against pushdown winning conditions is *undecidable*. In [5] a new winning condition for pushdown games was proposed, which declares a play winning if and only if along the play, the stack is *repeatedly bounded* (i.e. there is some stack depth n such that the stack was below depth n infinitely often). The main motivation for this winning condition was that it defined a class of plays that was in the $\mathbf{\Sigma_3}$ level of the Borel hierarchy (ω-regular winning conditions define only sets that are in the Boolean closure of $\mathbf{\Sigma_2}$). It was shown that solving these games was decidable. Note that for any pushdown game, if we label the push transitions, pop transitions, and internal transitions differently, then the set of repeatedly bounded plays is a visibly pushdown language.

Since visibly pushdown automata have a decidable model-checking problem and since the set of repeatedly bounded words is a VPL, a natural question arises: given a visibly pushdown game graph $\mathcal{G}$ and a VPL L describing the set of winning plays, is the game problem $(\mathcal{G}, L)$ decidable? The main result of this paper is that this problem is decidable and is 2EXPTIME-complete. Thus, the tractability of visibly pushdown languages extends to the game problem as well.

The main technical challenge in handling visibly pushdown games is that the specification automaton, which is a VPA, is not, in general, determinizable [2]. This prevents us from taking a product with the game graph to reduce it to a pushdown game with internal winning conditions. We invent a new kind of VPA, called *stair* VPA, in which the winning condition is interpreted only at certain points along the run, and the states met at other points are ignored. The i'th letter of a word belongs to this evaluation set if for no $j > i$, the stack depth at j is less than that at i. We then show that for every (nondeterministic) VPA, there exists an equivalent *deterministic* stair VPA. We take the product of the

game graph with the deterministic stair VPA, and show how pushdown games with stair winning conditions can be solved.

The above result yields a 3EXPTIME decision procedure for pushdown games against CARET specifications. However, this high complexity is not due to the context-free nature of the specification, as we show the surprising result that pushdown games against LTL specifications is already 3EXPTIME-hard. We also establish that solving pushdown games against nondeterministic VPA (or even nondeterministic Büchi automata) specifications is 2EXPTIME-hard.

Finally, we show that the class VPL is contained in the Boolean closure of $\mathbf{\Sigma_3}$, $B(\mathbf{\Sigma_3})$. This is one level higher than the class $B(\mathbf{\Sigma_2})$ which contains all regular ω-languages. As a consequence, we get an alternative proof that visibly pushdown automata cannot be determinized and also establish that visibly pushdown games are determined (i.e. from any position in the game, one of the players must have a winning strategy).

2 Preliminaries

For a finite set X we denote the set of finite words over X by X^*, the set of infinite words (ω-words) over X by X^ω, and the empty word by ε. For $\alpha \in X^* \cup X^\omega$ and $n \in \mathbb{N}$ we write $\alpha(n)$ for the nth letter in α and $\alpha\restriction_n$ for the prefix of length n of α, i.e., $\alpha\restriction_0 = \varepsilon$ and $\alpha\restriction_n = \alpha(0)\cdots\alpha(n-1)$ for $n \geq 1$.

A pushdown alphabet is a tuple $\widetilde{A} = \langle A_c, A_r, A_{\text{int}}\rangle$ that comprises three disjoint finite alphabets—A_c is a finite set of *calls*, A_r is a finite set of *returns*, and A_{int} is a finite set of *internal actions*. For any such $\widetilde{A}$, let $A = A_c \cup A_r \cup A_{\text{int}}$.

We define *visibly pushdown systems* over $\widetilde{A}$. Intuitively, the pushdown system is restricted such that it pushes onto the stack only when it reads a call, pops the stack only at returns, and does not use the stack on internal actions. The input hence controls the kind of operations permissible on the stack—however, there is no restriction on the symbols that can be pushed or popped.

Definition 1 (Visibly pushdown system [2]). *A visibly pushdown system (VPS) over $\langle A_c, A_r, A_{\text{int}}\rangle$ is a tuple $\mathcal{S} = (Q, Q_{in}, \Gamma, \Delta)$ where Q is a finite set of states, $Q_{in} \subseteq Q$ is a set of initial states, Γ is a finite stack alphabet that contains a special bottom-of-stack symbol $\perp$ and $\Delta \subseteq (Q \times A_c \times Q \times (\Gamma \setminus \{\perp\})) \cup (Q \times A_r \times \Gamma \times Q) \cup (Q \times A_{\text{int}} \times Q)$ is the transition relation.*

A stack is a nonempty finite sequence over Γ ending in the bottom-of-stack symbol $\perp$; let us denote the set of all stacks as $St = (\Gamma \setminus \{\perp\})^*.\{\perp\}$. A transition (q, a, q', γ), where $a \in A_c$ and $\gamma \neq \perp$, is a push-transition where on reading a, γ is pushed onto the stack and the control changes from state q to q'. Similarly, (q, a, γ, q') is a pop-transition where γ is read from the top of the stack and popped (if the top of stack is $\perp$, then it is read but not popped), and the control state changes from q to q'. Note that on internal actions, there is no stack operation.

The configuration graph of a VPS $\mathcal{S}$ is the graph $G_{\mathcal{S}} = (V_{\mathcal{S}}, E_{\mathcal{S}})$, where $V_{\mathcal{S}} = \{(q, \sigma) \mid q \in Q, \sigma \in St\}$, and $E_{\mathcal{S}}$ is the set that contains all triples $((q, \sigma), a, (q', \sigma')) \in V_{\mathcal{S}} \times A \times V_{\mathcal{S}}$, that satisfy the following:

[**Push**] If a is a call, then $\exists \gamma \in \Gamma$ such that $(q, a, q', \gamma) \in \Delta$ and $\sigma' = \gamma.\sigma$.
[**Pop**] If a is a return, then $\exists \gamma \in \Gamma$ such that $(q, a, \gamma, q') \in \Delta$ and either $\gamma \neq \bot$ and $\sigma = \gamma.\sigma'$, or $\gamma = \bot$ and $\sigma = \sigma' = \bot$.
[**Internal**] If a is an internal action, then $(q, a, q') \in \Delta$ and $\sigma = \sigma'$.

For a word $\alpha = a_1 a_2 a_3 \cdots$ in A^ω, a run of $\mathcal{S}$ on α is a sequence $\rho = (q_0, \sigma_0)(q_1, \sigma_1)(q_2, \sigma_2) \cdots \in V_{\mathcal{S}}^\omega$ of configurations, where $q_0 \in Q_{in}$, $\sigma_0 = \bot$ and $((q_i, \sigma_i), a, (q_{i+1}, \sigma_{i+1})) \in E_{\mathcal{S}}$ for every $i \in \mathbb{N}$.

A *visibly pushdown automaton* (VPA) over $\langle A_c, A_r, A_{\text{int}} \rangle$ is a tuple $\mathcal{M} = (Q, Q_{in}, \Gamma, \Delta, \Omega)$ where $(Q, Q_{in}, \Gamma, \Delta)$ is a VPS, and Ω is an acceptance condition. In a Büchi VPA, $\Omega = F \subseteq Q$ is a set of final states, while in a parity VPA, $\Omega : Q \to \mathbb{N}$.

For a run $\rho = (q_0, \sigma_0)(q_1, \sigma_1)(q_2, \sigma_2) \cdots$, we consider the set $\mathit{inf}(\rho) \subseteq Q$ which is the set of all states that occur in ρ infinitely often. A word $\alpha \in A^\omega$ is accepted by a Büchi VPA if there is a run ρ over α which infinitely often visits F, i.e., if $\mathit{inf}(\rho) \cap F \neq \emptyset$. A word $\alpha \in A^\omega$ is accepted by a parity VPA if there is a run ρ over α such that the minimal color visited infinitely often by ρ is even, i.e., if the minimal color in $\Omega(\mathit{inf}(\rho)) = \{\Omega(q) \mid q \in \mathit{inf}(\rho)\}$ is even. The language $L(\mathcal{M})$ of a VPA $\mathcal{M}$ is the set of words accepted by $\mathcal{M}$.

A VPA is deterministic if it has a unique initial state q_{in}, and for each input letter and configuration there is at most one successor configuration. For deterministic VPAs we denote the transition relation by δ instead of Δ and write $\delta(q, a) = (q', \gamma)$ instead of $(q, a, q', \gamma) \in \delta$ if $a \in A_c$, $\delta(q, a, \gamma) = q'$ instead of $(q, a, \gamma, q') \in \delta$ if $a \in A_r$, and $\delta(q, a) = q'$ instead of $(q, a, q') \in \delta$ if $a \in A_{\text{int}}$.

Infinite two-player games. Let A be a finite alphabet. A *game graph* $\mathcal{G}$ over A is a graph $\mathcal{G} = (V, V_{\text{E}}, V_{\text{A}}, E)$ where (V, E) is a deterministic graph with edges labeled with letters of A (i.e. $E \subseteq V \times A \times V$ such that if $(v, a, v_1), (v, a, v_2) \in E$, then $v_1 = v_2$), and $(V_{\text{E}}, V_{\text{A}})$ partitions V between two players, Eve and Adam. An infinite two-player game is a pair $\mathbb{G} = (\mathcal{G}, \Omega)$, where the winning condition Ω can be of two kinds: An *internal winning condition* Ω is a subset of V^ω and an *external winning condition* Ω is a subset of A^ω.

The players, Eve and Adam, play in $\mathbb{G}$ by moving a token between positions. A *play* from some initial node v_0 proceeds as follows: the player owning v_0 moves the token to some vertex v_1 along an edge of the form $e_0 = (v_0, a_0, v_1) \in E$. Then the player owning v_1 moves the token to v_2 along an edge $e_1 = (v_1, a_1, v_2)$, and so on, forever. If one of the players cannot make a move, the other player wins. Otherwise, the play is an infinite sequence $\lambda = v_0 a_0 v_1 a_1 \cdots \in (V.A)^\omega$ in G. For internal winning conditions, Eve wins λ if $v_0 v_1 v_2 \cdots \in \Omega$, and Adam wins it otherwise. If Ω is an external winning condition, Eve wins λ if $a_0 a_1 a_2 \cdots \in \Omega$, and Adam wins it otherwise. A *partial play* is any prefix of a play.

A *strategy* for Eve is a function assigning to any partial play ending in some node in $v \in V_{\text{E}}$ an edge $(v, a, v') \in E$. Eve *respects* a strategy f during some play $\lambda = v_0 a_0 v_1 a_1 \cdots$ if for any $i \geq 0$ such that $v_i \in V_{\text{E}}$, $(v_i, a_{i+1}, v_{i+1}) = f(v_0 a_0 v_1 a_1 \cdots v_i)$. Finally, a strategy f is said to be *winning* from some position v, if any play starting from v where Eve respects f is winning for her.

A *visibly pushdown game* $\mathcal{H} = (\mathcal{S}, Q_{\mathrm{E}}, Q_{\mathrm{A}}, \mathcal{M})$ consists of a VPS $\mathcal{S}$, a VPA $\mathcal{M}$ (both over a common pushdown alphabet $\widetilde{A}$), and a partition $\langle Q_{\mathrm{E}}, Q_{\mathrm{A}} \rangle$ of the state set Q of $\mathcal{S}$. $\mathcal{H}$ defines the game $\mathbb{G}_{\mathcal{H}} = (\mathcal{G}, \Omega)$, where $\mathcal{G} = (V, V_{\mathrm{E}}, V_{\mathrm{A}}, E)$, (V, E) is the configuration graph of $\mathcal{S}$, $V_{\mathrm{E}} = \{(q, \sigma) \mid q \in Q_{\mathrm{E}}\}$, and $V_{\mathrm{A}} = \{(q, \sigma) \mid q \in Q_{\mathrm{A}}\}$. The set Ω is the external winning condition $\Omega = L(\mathcal{M})$.

We can now state the main problem we address in this paper: Given a visibly pushdown game $\mathcal{H} = (\mathcal{S}, Q_{\mathrm{E}}, Q_{\mathrm{A}}, \mathcal{M})$ and a state p_{in} of $\mathcal{S}$, is there a strategy for Eve that is winning for her from the position $(p_{in}, \bot)$, in the game $\mathbb{G}_{\mathcal{H}}$?

3 Deterministic Stair VPAs

Visibly pushdown automata over ω-words cannot be determinized [2]. In this section, in order to obtain a determinization theorem, we propose a new mode of acceptance for VPAs. Instead of evaluating the acceptance condition on the whole run, we evaluate it only on a subsequence of the run. This subsequence is obtained by discarding those configurations for which a future configuration of smaller stack height exists. The sequence thus obtained is non-decreasing with respect to the stack height, and hence we dub VPAs using this mode of acceptance as *stair* VPAs (denoted STVPA). The main theorem of this section is that for every nondeterministic Büchi VPA, we can effectively construct an equivalent deterministic parity STVPA.

For $Y \subseteq \mathbb{N}$ and $\rho \in X^\omega$ (for some set X) we define the subsequence $\rho|_Y \in X^* \cup X^\omega$ of ρ induced by Y as follows. Let $n_0 < n_1 < n_2 < \cdots$ be an ascending enumeration of the elements in Y. Then $\rho|_Y = \rho(n_0)\rho(n_1)\rho(n_2)\cdots$.

For $w \in A^*$ we define the stack height $sh(w)$ inductively by $sh(\varepsilon) = 0$ and

$$sh(ua) = \begin{cases} sh(u) & \text{if } a \in A_{\mathrm{int}}, \\ sh(u) + 1 & \text{if } a \in A_c, \\ \max\{sh(u) - 1, 0\} & \text{if } a \in A_r. \end{cases}$$

For $\alpha \in A^\omega$ define $\mathit{Steps}_\alpha = \{n \in \mathbb{N} \mid \forall m \geq n \,:\, sh(\alpha\restriction_m) \geq sh(\alpha\restriction_n)\}$. Note that Steps_α is infinite for each $\alpha \in A^\omega$.

Let L_{mwm} denote the set of all *minimally well-matched words*—the words of the form $cwr \in A^*$, where the last letter $r \in A_r$ is the matching return for the first letter $c \in A_c$ (formally, $c \in A_c$, $r \in A_r$, $sh(w) = 0$ and for any prefix w' of w, $sh(cw') > 0$).

For any word $\alpha \in A^\omega$, we can group maximal subwords of α which are in L_{mwm}, and get a unique factorization $\alpha = w_1 w_2 \ldots$ where each $w_i \in L_{\mathrm{mwm}} \cup A$. It is easy to see that if $w_i = c$, for some $c \in A_c$, then there is no $j > i$ such that $w_j = r$, for some $r \in A_r$. In fact, the points at which the word factorizes is exactly Steps_α, i.e. $n \in \mathit{Steps}_\alpha$ iff $\exists i \geq 0 : |w_1 \ldots w_i| = n$.

To define acceptance for STVPAs, we evaluate the acceptance condition at the subsequence $\rho|_{\mathit{Steps}_\alpha}$ for any run ρ on α, i.e. at the positions after each prefix $w_1 \ldots w_i$, where $i \in \mathbb{N}$.

Definition 2 (Stair VPA). *A (nondeterministic) stair* VPA (StVPA) $\mathcal{M} = (Q, Q_{in}, \Gamma, \Delta, \Omega)$ *over* $\langle A_c, A_r, A_{\text{int}}\rangle$ *has the same components as a* VPA. *A word* $\alpha \in A^\omega$ *is accepted by* $\mathcal{M}$ *if there is a run* ρ *of* $\mathcal{M}$ *on* α *such that* $\rho|_{Steps_\alpha}$ *satisfies the acceptance condition* Ω *of* $\mathcal{M}$. *The language accepted by* $\mathcal{M}$ *is* $L(\mathcal{M}) = \{\alpha \in A^\omega \mid \mathcal{M} \text{ accepts } \alpha\}$.

Example 1. Let $L_{\text{rb}} = \{\alpha \in A^\omega \mid \exists \ell\, \forall m\, \exists n > m \;:\; sh(\alpha\restriction_n) = \ell\}$ (with $A_{\text{int}} = \emptyset$, $A_r = \{r\}$, and $A_c = \{c\}$) be the set of all repeatedly bounded words. As shown in [2] there is no deterministic VPA for this language. Now consider the parity StVPA $\mathcal{M}_{\text{rb}}$ with states q_1, q_2, initial state q_1, stack alphabet $\Gamma = \{\gamma, \bot\}$, coloring function $\Omega(q_1) = 1$, $\Omega(q_2) = 2$, and transition function $\delta(q_1, c) = \delta(q_2, c) = (q_1, \gamma)$ and $\delta(q_1, r, \gamma) = \delta(q_2, r, \gamma) = \delta(q_1, r, \bot) = \delta(q_2, r, \bot) = q_2$. For a run ρ of this StVPA the sequence $\rho|_{Steps_\alpha}$ contains infinitely many q_1 iff the input contains infinitely many unmatched calls and thus $L(\mathcal{M}_{\text{rb}}) = L_{\text{rb}}$.

We aim at proving that for each nondeterministic Büchi VPA $\mathcal{M}$ there is an equivalent deterministic parity StVPA D. Let $\alpha \in A^\omega$ and let the factorization of α be $\alpha = w_1 w_2 \ldots$. A stair VPA reading α can refer to the states after each w_i only. In order to capture the way $\mathcal{M}$ acts on a subword w_i, we use *summary information* which, intuitively, describes all possible transformations $\mathcal{M}$ can undergo when reading the word w_i. For this purpose let $\mathcal{M} = (Q, Q_{in}, \Gamma, \Delta, F)$ and set $\mathcal{T}_Q = 2^{Q \times \{0,1\} \times Q}$. The transformation $T_{w_i} \in \mathcal{T}_Q$ induced by w_i is defined as follows: $(q, f, q') \in T_{w_i}$ iff there is a run of $\mathcal{M}$ on w_i leading from $(q, \bot)$ to (q', σ), for some $\sigma \in St$, with $f = 1$ iff this run meets some state in F. Note that the initial stack content does not matter if $w_i \in A_c \cup A_{\text{int}} \cup L_{\text{mwm}}$, and if $w_i \in A_r$, we know that when w_i occurs in α, the stack must be empty.

Now consider the sequence $\tau_\alpha = T_{w_1} T_{w_2} \ldots \in \mathcal{T}_Q^\omega$. $\mathcal{M}$ accepts α iff we can string together a consistent run using the summaries in τ_α such that it visits F infinitely often. Formally, a word $\tau \in \mathcal{T}_Q^\omega$ is *good* if there exists $\rho \in Q^\omega$ such that $\rho(0) \in Q_{in}$ and for all $i \in \mathbb{N}$, $(\rho(i), f_i, \rho(i+1)) \in \tau(i)$, for some $f_i \in \{0, 1\}$, where $f_i = 1$ for infinitely many $i \in \mathbb{N}$. Then it is easy to see that $\alpha \in L(\mathcal{M})$ iff τ_α is good. Note that the set of all good words over $\mathcal{T}_Q$ is in fact a regular ω-language over the alphabet $\mathcal{T}_Q$. Hence we can build a deterministic parity automaton $S_\mathcal{T} = (S, s_{in}, \delta, \Omega)$ which accepts the set of all good words. Moreover, $S_\mathcal{T}$ can be constructed such that $S = 2^{\mathcal{O}(|Q| \cdot \log |Q|)}$ [13].

We can also show that the summary information can be generated by a deterministic VPS. Formally, there is a deterministic VPS C with output such that on reading any finite word w, if the factorization of w is $w'_1 \ldots w'_k$, C outputs the transformation $T_{w'_k}$ on its last transition. Such a VPS C is easy to construct: the state-space of C is $\mathcal{T}_Q$ with initial state $Id_Q = \{(q, 0, q) \mid q \in Q\}$. On reading an internal action $a \in A_{\text{int}}$ (or on reading a return when the stack is empty), C updates its state from T to $T \circ T_a$ and outputs T_a; on reading a call $c \in A_c$, it pushes c and the current state T onto the stack, updates the state to Id_Q, and outputs T_c; on reading a return $r \in A_r$ when the stack is nonempty, it pops T' and $c \in A_c$, updates its state from T to $T' \circ \bigcup_{\gamma \in \Gamma}(T_{c,\gamma} \circ T \circ T_{r,\gamma})$, and outputs $\bigcup_{\gamma \in \Gamma}(T_{c,\gamma} \circ T \circ T_{r,\gamma})$. Here, $T_1 \circ T_2$ is defined to be the set of all triples

(q, f, q') such that there are some elements $(q, f_1, q_1) \in T_1$, $(q_1, f_2, q') \in T_2$ and $f = \max\{f_1, f_2\}$. The transformation $T_{c,\gamma}$ (resp. $T_{r,\gamma}$) is the one induced by the transitions pushing γ on reading c (resp. popping γ on reading r).

We are now ready to construct the deterministic parity StVPA D accepting $L(\mathcal{M})$. The state-space of D is $\mathcal{T}_Q \times S$, and we will construct D such that after reading any finite word w with factorization $w = w'_1, \ldots w'_k$, the second component of D's state is the state which $S_\mathcal{T}$ would reach on the word $T_{w'_1} \ldots T_{w'_k}$. D inherits the parity condition from $S_\mathcal{T}$ and it is easy to see that the above property ensures that D accepts $L(\mathcal{M})$.

D simulates the VPS C on the first component and the second component is updated using the outputs of C. In addition to the information stored on the stack by C, when reading a call symbol $c \in A_c$, D also pushes onto the stack the state it was in before the call symbol was read. When D reads a return symbol and the stack is not empty, the second component needs to be updated to $\delta(s, T)$ where s is the state $S_\mathcal{T}$ was in before it read the call corresponding to the current return, and T is the summary of the segment from the corresponding call to the current return. The state s is available on the top of the stack (since D had pushed it at the corresponding call) and T corresponds to the output of C; hence D can update the second component appropriately. We have:

Theorem 1. *For each nondeterministic Büchi* VPA $\mathcal{M}$ *over* A *there exists a deterministic parity* StVPA D *such that* $L(\mathcal{M}) = L(D)$. *Moreover, we can construct* D *such that it has* $2^{\mathcal{O}(|Q|^2)}$ *states, where* Q *is the state-space of* $\mathcal{M}$.

As Theorem 1 shows, evaluating the acceptance condition on $\rho|_{Steps_\alpha}$ instead of ρ increases the expressive power of deterministic VPAs. A nondeterministic VPA can guess the positions of $Steps_\alpha$ (and verify its correctness), and hence stair acceptance does not change the expressive power of nondeterministic VPAs.

Theorem 2. *For each nondeterministic parity* StVPA $\mathcal{M}$ *one can construct a nondeterministic Büchi* VPA $\mathcal{M}'$ *such that* $L(\mathcal{M}) = L(\mathcal{M}')$.

4 Games

In this section, our main aim is to prove that the problem of solving visibly pushdown games as stated at the end of Section 2 is in 2EXPTIME. Our first step is to internalize the winning condition $\mathcal{M}$ by transforming it to a deterministic stair VPA and then taking its product with the game graph defined by $\mathcal{H}$. This results in a game with a stair parity winning condition, which we then solve.

A *stair parity game* $\mathcal{ST} = (\mathcal{S}, Q_\mathrm{E}, Q_\mathrm{A}, col)$ consists of a VPS $\mathcal{S} = (Q, Q_{in}, \Gamma, \Delta)$, a partition $\langle Q_\mathrm{E}, Q_\mathrm{A}\rangle$ of Q, and a coloring function $col : Q \to \mathbb{N}$. The game defined by $\mathcal{ST}$ is $\mathbb{G}_{\mathcal{ST}} = (\mathcal{G}, \Omega)$ with $\mathcal{G} = (V, V_\mathrm{E}, V_\mathrm{A}, E)$, where (V, E) is the configuration graph of $\mathcal{S}$, $V_\mathrm{E} = \{(p, \sigma) \mid p \in Q_\mathrm{E}\}$, and $V_\mathrm{A} = \{(p, \sigma) \mid p \in Q_\mathrm{A}\}$. The set Ω is the internal winning condition $\Omega = \{\lambda \in V^\omega \mid \min_{col}(\lambda|_{Steps_\lambda}) \text{ is even}\}$ where $\min_{col}(\beta) = \min\{i \mid \exists^\infty n \ s.t. \ col(\beta(n)) = i\}$. Here, $Steps_\lambda$ is the natural adaption of the definition of $Steps_\alpha$ to sequences of configurations, i.e, $Steps_\lambda = \{n \in \mathbb{N} \mid \forall m \geq n \ |\lambda(m)| \geq |\lambda(n)|\}$.

Note that the labeling of the edges in a stair parity game does not matter and, in the sequel, we will ignore it.

To transform a visibly pushdown game $\mathcal{H} = (\mathcal{S}, Q_{\mathrm{E}}, Q_{\mathrm{A}}, \mathcal{M})$ into a stair parity game let D be some deterministic StVPa such that $L(D) = L(\mathcal{M})$. Since $\mathcal{S}$ and D are over the same pushdown alphabet $\widetilde{A}$, we can take the synchronized product $\mathcal{S} \otimes D$ to get a pushdown system $\mathcal{S}'$ (ignoring the acceptance condition). We then have a stair parity game $\mathcal{ST} = (\mathcal{S}', Q'_{\mathrm{E}}, Q'_{\mathrm{A}}, col)$, where the partition of the state-space is inherited from $\mathcal{H}$ and the coloring function is inherited from the coloring function of D. Since D is deterministic one can easily show the following proposition, where q_{in} denotes the initial state of D.

Proposition 1. *Let $p_{in} \in Q$. Then $(p_{in}, \bot)$ is winning for Eve in $\mathbb{G}_{\mathcal{H}}$ if and only if $((p_{in}, q_{in}), \bot)$ is winning for Eve in $\mathbb{G}_{\mathcal{ST}}$.*

Now we explain how to adapt the classical techniques for pushdown parity games and its variants [15, 4, 11] in order to solve stair parity games.

Let $\mathcal{ST} = (\mathcal{S}, Q_{\mathrm{E}}, Q_{\mathrm{A}}, col)$ be a stair parity game, where $\mathcal{S} = (Q, Q_{in}, \Gamma, \Delta)$ and let $\mathcal{G} = (V, V_{\mathrm{E}}, V_{\mathrm{A}}, E)$ be the associated game graph. We construct a *finite* game graph $\overline{\mathcal{G}}$ with a parity winning condition, such that Eve has a winning strategy in $\mathcal{G}$ iff she has a winning strategy in $\overline{\mathcal{G}}$. Intuitively, in $\overline{\mathcal{G}}$, we keep track of only the control state and the symbol on the top of the stack. The interesting aspect of the game is when it is in a control state p with top-of-stack γ, and the player owning p wants to push a letter γ' onto the stack. For every strategy of Eve there is a certain set of possible (finite) continuations of the play that will end with popping this γ' symbol from the stack. We require Eve to declare the set R of all states the game can be in after the popping of γ' along these plays.

Adam now has two choices—he can either continue the game by pushing γ' onto the stack and updating the state (we call this a *pursue* move), or he can pick some state $p'' \in R$ and continue from that state, leaving γ on the top of the stack (we call this a *jump* move). If he does a pursue move, then he remembers R and if there is a pop-transition on γ' later on in the play, the play stops right there and Eve is declared the winner if and only if the resulting state is in R.

The crucial point to note is that the *jump* transitions along infinite plays in $\overline{\mathcal{G}}$ (i.e. plays that never meet a pop-transition with the stack being non-empty) essentially skip words of L_{mwm}, and hence the play really corresponds to evaluating a play λ in the pushdown game at $Steps_{\lambda}$. Therefore the stair parity condition gets evaluated along the play and ensures correctness of the reduction.

Let us now describe the construction more precisely. The main nodes of $\overline{\mathcal{G}}$ are tuples in $Q \times \Gamma \times 2^{Q}$. A node (p, γ, R) has color $col(p)$ and belongs to Eve iff $p \in Q_{\mathrm{E}}$. Intuitively, a node (p, γ, R) denotes that the current state of $\mathcal{S}$ is p, γ is the symbol on the top of the stack, and R is the current commitment Eve has made, i.e. Eve has claimed that if a pop-γ transition is executed, then the resulting state will be in R. The starting node is $(p_{in}, \bot, \emptyset)$.

In order to simulate an internal-transition $(p, p') \in \Delta$, we have edges of the form $(p, \gamma, R) \to (p', \gamma, R)$ in $\overline{\mathcal{G}}$. Also, if the stack is empty, pop-transitions are handled like internal transitions: if $(p, \bot, p') \in \Delta$, then there is an edge $(p, \bot, R) \to (p', \bot, R)$ in $\overline{\mathcal{G}}$.

Pop-transitions are not simulated but are represented in $\overline{\mathcal{G}}$ by edges to a vertex tt (winning for Eve) and a vertex ff (winning for Adam) to verify the claims made by Eve. Recall that in (p, γ, R) the set R represents the claim of Eve that on a pop-γ transition the next state will be in R. Hence, in $\overline{\mathcal{G}}$ there is an edge from (p, γ, R) to tt if there is $p' \in R$ and a pop-transition $(p, \gamma, p') \in \Delta$. If p belongs to Eve, then this transition can be used by Eve to win the game because she was able to prove that her claim was correct. If there is a pop-transition $(p, \gamma, p') \in \Delta$ with $p' \notin R$, then there is an edge from (p, γ, R) to ff, which can be used by Adam to win (if p belongs to Adam) since Eve made a false claim.

The simulation of a push-transition takes place in several steps. For a node (p, γ, R) the player owning p first picks a particular push-transition (p, p', γ') by moving to the node $(p, \gamma, R, p', \gamma')$, which belongs to Eve. Then Eve proposes a set $R' \subseteq Q$ containing the states that she claims to be reached if γ' gets eventually popped. She does this by moving to the node $(p, \gamma, R, p', \gamma', R')$, which belongs to Adam. Now, Adam has two kinds of choices. He can do a *jump* move by picking a state $p'' \in R'$ and move to the node (p'', γ, R). Or he can do a *pursue* move by moving to the node (p', γ', R').

If $\overline{\mathbb{G}}$ denotes the parity game played on $\overline{\mathcal{G}}$, we get the following result which can be shown using similar methods as, e.g., in [15, 4, 11].

Theorem 3. *Let $p_{in} \in Q$. Eve has a winning strategy from $(p_{in}, \bot)$ in the pushdown game $\mathbb{G}_{\mathcal{ST}}$ if and only if she has a winning strategy in $\overline{\mathbb{G}}$ from $(p_{in}, \bot, \emptyset)$. In addition, one can effectively build pushdown strategies for both players in $\mathbb{G}_{\mathcal{ST}}$.*

As a corollary of Theorem 3, Proposition 1, and the fact that the transformation from Proposition 1 preserves pushdown strategies, we have the following:

Corollary 1. *The problem of deciding the winner in a visibly pushdown game is in* 2EXPTIME *and pushdown strategies can be effectively built for both players.*

It is a well known result that there always exists memoryless winning strategies in parity games [6, 17]. Nevertheless, it is not the case for the preceding winning conditions:

Proposition 2. *There exist a stair parity (*resp. *visibly) pushdown game and a configuration winning for Eve such that any winning strategy for Eve from this position requires infinite memory.*

Visibly pushdown games are solvable in 2EXPTIME, as we showed above. Let us now consider pushdown games where the alphabet A is a subset of $2^{\mathcal{P}}$ where $\mathcal{P}$ is a finite set of propositions. CARET is a temporal logic that can express a subclass of context-free languages which is contained in VPL [1, 2]. From constructions in [1], it follows that for every CARET formula φ over $2^{\mathcal{P}}$, and a partition $\widetilde{A}$ of $2^{\mathcal{P}}$ into calls, returns, and internal actions, we can construct a Büchi visibly pushdown automaton of size $2^{O(|\varphi|)}$ over $\widetilde{A}$ which accepts the precise set of strings that satisfy φ. Hence, it follows that solving visibly pushdown games against CARET specifications is in 3EXPTIME.

However, this high complexity is not due to the pushdown nature of the specification nor due to the fact that we are dealing with ω-length plays. If we consider pushdown games against an LTL specification φ, we can solve this by first constructing a nondeterministic Büchi automaton accepting the models of φ and then constructing an equivalent deterministic parity automaton for it (resulting in an automaton whose size is doubly exponential in φ). Then, we can take the product of the pushdown game and this automaton, and solve the resulting parity pushdown game in exponential time [15]. The whole procedure works in 3EXPTIME. By a reduction from the word problem for alternating doubly exponential space bounded Turing machines one can show that this is a lower bound as well:

Theorem 4. *Given a pushdown game and an* LTL *formula, checking whether Eve has a winning strategy is* 3EXPTIME*-complete.*

We also establish the exact complexity of the following pushdown game problems:

Theorem 5.

- *Given a pushdown game and a* CARET *formula, checking whether Eve has a winning strategy is* 3EXPTIME*-complete.*
- *Given a pushdown game and a nondeterministic Büchi automaton, checking whether Eve has a winning strategy is* 2EXPTIME*-complete.*
- *Given a visibly pushdown game graph and a nondeterministic Büchi* VPA*, checking whether Eve has a winning strategy is* 2EXPTIME*-complete.*

5 Topological Complexity

It is well known that the class of regular ω-languages is contained in the Boolean closure of the second level of the Borel hierarchy. Our goal is to show that this topological complexity is increased only by one level when we pass to visibly pushdown languages, i.e., we show that the class of visibly pushdown languages is contained in the Boolean closure of the third level of the Borel hierarchy. For more details on the definitions and results used in this section we refer the reader to [7] for set-theory in general and to [12] for results related to ω-languages.

For a set X we consider X^ω as a topological space with the Cantor topology. The open sets of X^ω are those of the form $U \cdot X^\omega$ for $U \subseteq X^*$. A set $L \subseteq X^\omega$ is closed if its complement $L^- = X^\omega \setminus L$ is open.

To define the finite levels of the Borel hierarchy we start with the class $\mathbf{\Sigma_1}$ of open sets. For each $n \geq 1$, $\mathbf{\Pi_n}$ is the class of complements of $\mathbf{\Sigma_n}$-sets and $\mathbf{\Sigma_{n+1}}$ is the class of countable unions of $\mathbf{\Pi_n}$-sets. By $B(\mathbf{\Sigma_n})$ we denote the class of finite Boolean combinations of $\mathbf{\Sigma_n}$-sets (using union, intersection, and complement).

For $L_1 \subseteq X_1^\omega, L_2 \subseteq X_2^\omega$ we say L_1 reduces continuously to L_2 if there is a continuous mapping $\varphi : X_1^\omega \to X_2^\omega$ such that $\varphi^{-1}(L_2) = L_1$, i.e., $\alpha \in L_1$ iff $\varphi(\alpha) \in L_2$ for all $\alpha \in X_1^\omega$. A language $L \subseteq X^\omega$ is called $\mathbf{\Sigma_n}$-complete if it is in $\mathbf{\Sigma_n}$ and every $K \in \mathbf{\Sigma_n}$ continuously reduces to L. The definition of $\mathbf{\Pi_n}$-complete sets is analogous.

We show the result that any VPL L belongs to $B(\mathbf{\Sigma_3})$ by using the model of stair VPA introduced in Section 3. Let $L \subseteq A^\omega$ be a VPL and let $\mathcal{M} = (Q, q_{in}, \Gamma, \delta, \Omega)$ be a deterministic parity stair VPA with $L(\mathcal{M}) = L$. To show that L is in $B(\mathbf{\Sigma_3})$, we define for each $q \in Q$ the language L_q containing all the words α for which the run of $\mathcal{M}$ on α infinitely often visits q on positions from Steps_α, and show that L_q belongs to $\mathbf{\Pi_3}$. The language L itself can be written as a finite Boolean combination of the sets L_q corresponding to the definition of the parity acceptance condition: $\alpha \in A^\omega$ is in L iff $\alpha \in L_q$ for some q with $\Omega(q)$ even and $\alpha \notin L_{q'}$ for all q' with $\Omega(q') < \Omega(q)$.

For the definition of L_q we will use the following sets of finite words.

- For each $q \in Q$, let $U_q \subseteq A^*$ be the set of all words w such that the run of $\mathcal{M}$ on w ends in a configuration with state q.
- Let $U_{\mathrm{mr}} = (A_c \cup A_{\mathrm{int}} \cup L_{\mathrm{mwm}})^*$ be the set of all words without unmatched returns.
- Let $U_0 = (A_r \cup A_{\mathrm{int}} \cup L_{\mathrm{mwm}})^*$ be the set of all words of stack height 0.

We describe L_q by stating that for each position $m \in \mathbb{N}$ there is a position $n > m$ that is in Steps_α and the prefix of α up to position n is in U_q. The only difficulty is to express that position n is in the set Steps_α. For this we distinguish two cases (which are not mutually exclusive). Position n is in Steps_α if $\alpha{\restriction_n}$ has stack height 0 or if the suffix of α starting from position n does not contain any unmatched returns. Formally, for $\alpha \in A^\omega$ and $n \in \mathbb{N}$ we get that $n \in \mathit{Steps}_\alpha$ and the run of $\mathcal{M}$ on $\alpha{\restriction_n}$ ends in a configuration with state q iff α is in the set

$$L_{q,n} = [(U_q \cap A^n).A^\omega] \cap \left[(U_0 \cap A^n).A^\omega \cup \left(\bigcap_{n'>n} (A^n.U_{\mathrm{mr}} \cap A^{n'}).A^\omega \right) \right].$$

The basic sets involved in this definition are of the form $U.A^\omega$ for U finite (since we always intersect with the set of words up to a certain length). These sets are open as well as closed. Since the class of closed sets is closed under countable intersections and finite unions we obtain that $L_{q,n}$ is closed for each q and n.

By adding the quantifications for m and n we obtain the following definition of L_q: $L_q = \bigcap_{m \in \mathbb{N}} \bigcup_{n>m} L_{q,n}$. It directly follows from the definition that L_q is in $\mathbf{\Pi_3}$ and hence we obtain the following theorem.

Theorem 6. *The class of* ω-VPL*s is contained in* $B(\mathbf{\Sigma_3})$.

One should note that there are nondeterministic Büchi VPAs accepting $\mathbf{\Sigma_3}$-complete sets. The language L_{rb} from Example 1 is shown to be $\mathbf{\Sigma_3}$-complete in [5]. The complement of this language is $\mathbf{\Pi_3}$-complete and is also a VPL (since visibly pushdown languages are closed under complement).

There are no complete sets for the class $B(\mathbf{\Sigma_3})$ but it is not difficult to see that there are VPLs that are true $B(\mathbf{\Sigma_3})$-sets in the sense that they are neither in $\mathbf{\Sigma_3}$ nor in $\mathbf{\Pi_3}$. A simple way to define such a language is to consider an alphabet A with priorities assigned to the letters, i.e., there are k calls, k internal actions, and k returns, respectively, and they are assigned numbers from 1 to k. If we

define L to be the language containing all α such that $\alpha|_{Steps_\alpha}$ satisfies the parity condition w.r.t. the numbers assigned to the letters, then it is not difficult to see that L is neither in $\mathbf{\Sigma_3}$ nor in $\mathbf{\Pi_3}$. But obviously L can be accepted by a StVPA that moves on each letter to a state with the corresponding priority.

Furthermore, let us note that languages accepted by deterministic VPAs are in $B(\mathbf{\Sigma_2})$. The proof is similar to the one showing that regular ω-languages are in $B(\mathbf{\Sigma_2})$ [12]. From this result we obtain an alternative proof that the language L_{rb} cannot be accepted by a deterministic VPA, since L_{rb} is $\mathbf{\Sigma_3}$-complete. Finally, the results of this section imply that games with a VPL winning condition are determined because games with Borel winning conditions are determined [8].

References

1. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *TACAS'04*, volume 2988 of *LNCS*, pages 467–481. Springer, 2004.
2. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC '04*, 2004.
3. T. Ball and S. Rajamani. Bebop: A symbolic model checker for boolean programs. In *SPIN 2000*, volume 1885 of *LNCS*, pages 113–130. Springer, 2000.
4. A. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with the unboundedness and regular conditions. In *Proceedings of FSTTCS'03*, volume 2914 of *LNCS*, pages 88–99. Springer, 2003.
5. T. Cachat, J. Duparc, and W. Thomas. Solving pushdown games with a Σ_3 winning condition. In *CSL'02*, volume 2471 of *LNCS*, pages 322–336. Springer, 2002.
6. E.A. Emerson, C.S. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *CAV '93*, volume 697 of *LNCS*, pages 385–396. Springer, 1993.
7. A.S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate texts in mathematics*. Springer Verlag, 1994.
8. D. A. Martin. Borel Determinacy. *Annals of Mathematics*, 102:363–371, 1975.
9. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symposium on Principles of Programming Languages*, Austin, January 1989.
10. T. Reps, S. Horwitz, and S. Sagiv. Precise interprocedural dataflow analysis via graph reachability. In *Proc. of ACM Symp. POPL*, pages 49–61, 1995.
11. O. Serre. Games with winning conditions of high borel complexity. In *Proceedings of ICALP'04*, volume 3142 of *LNCS*, pages 1150–1162. Springer, 2004.
12. L. Staiger. *Handbook of Formal Language Theory*, volume III, chapter ω-Languages, pages 339–387. Springer, 1997.
13. W. Thomas. *Handbook of Formal Language Theory*, volume III, chapter Languages, Automata, and Logic, pages 389–455. Springer, 1997.
14. W. Thomas. A short introduction to infinite automata. In *Proceedings of DLT '01*, volume 2295 of *LNCS*, pages 130–144. Springer, 2002.
15. I. Walukiewicz. Pushdown processes: Games and model checking. *Information and Computation*, 164(2), January 2001.
16. I. Walukiewicz. A landscape with games in the background. In *Proceedings of LICS'04, Invited talk*, 2004. To appear.
17. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *TCS*, 200(1-2):135–183, 1998.