Winning Regions of Pushdown Parity Games: A Saturation Method

Matthew Hague and C.-H. Luke Ong

Oxford University Computing Laboratory

Abstract. We present a new algorithm for computing the winning region of a parity game played over the configuration graph of a pushdown system. Our method gives the first extension of the saturation technique to the parity condition. Finite word automata are used to represent sets of pushdown configurations. Starting from an initial automaton, we perform a series of automaton transformations to compute a fixed-point characterisation of the winning region. We introduce notions of under-approximation (soundness) and over-approximation (completeness) that apply to automaton transitions rather than runs, and obtain a clean proof of correctness. Our algorithm is simple and direct, and it permits an optimisation that avoids an immediate exponential blow up.

1 Introduction

Pushdown systems — finite-state transition systems equipped with a stack — are an old model of computation that have recently enjoyed renewed interest from the software verification community. They accurately model the control flow of first-order recursive programs [7] (such as C and Java), and lend themselves readily to algorithmic analysis. Pushdown systems have played a key rôle in the automata-theoretic approach to software model checking [1,5,10,14]. Considerable progress has been made in the implementation of scalable model checkers of pushdown systems. These tools (e.g. Bebop [11] and Moped [10]) are an essential back-end component of such model checkers as SLAM [12].

The modal mu-calculus is a highly expressive language for describing properties of program behaviour (all standard temporal logics in verification are embeddable in it). In a seminal paper [3] at CAV 1996, Walukiewicz showed that *local* modal mu-calculus model checking of pushdown systems, or equivalently [4] the solution of *pushdown parity games* (i.e. parity games over the configuration graphs of pushdown systems), is EXPTIME-complete. His method reduces pushdown parity games to finite parity games by a kind of powerset construction, which is immediately exponential in size. Whilst *local* model checking asks if a designated state (of a pushdown system) satisfies a given property, *global* model checking computes a finite representation of the set of states satisfying the property. The latter is equivalent to computing Éloïse's winning region of a pushdown parity game, which is the problem that we have set ourselves here. It is worth noting that global model checking used to be the norm in verification (CTL and many symbolic model checkers still perform global model checking). While local model checking can be expected to have better complexity, global model checking is important when repeated checks are required (because tests on the representing automata

M. Bravetti and G. Zavattaro (Eds.): CONCUR 2009, LNCS 5710, pp. 384–398, 2009.

tend to be comparatively cheap), or where the model checking is only a component of the verification process.

Related work. Cachat [13] and Serre [9] have independently generalised Walukiewicz' algorithm to provide solutions to the global model-checking problem: they use the local model-checking algorithm as an oracle to guide the construction of the automaton recognising the winning region. An alternative approach, introduced by Piterman and Vardi [8], uses two-way alternating tree automata to navigate a tree representing all possible stacks: after several reductions, including the complementation of Büchi automata, an automaton accepting the winning region can be constructed.

At CONCUR 1997, Bouajjani *et al.* [1], and, independently, Finkel *et al.* [2] (at INFINITY 1997), introduced a *saturation* technique for global model-checking reachability properties of pushdown systems. From a finite-word automaton recognising a given configuration-set $\mathcal{C}$, they perform a backwards-reachability analysis. By iteratively adding new transitions to the automaton, the set of configurations that can reach some configuration in $\mathcal{C}$ is constructed. Since the number of new transitions is bounded, the iterative process terminates. This approach underpins the acclaimed Moped tool.

Contributions. This paper presents a new algorithm for computing Éloïse's winning region of a pushdown parity game. We represent (regular) configuration sets as alternating multi-automata [1]. Using a modal mu-calculus formula that defines the winning region as a guide, our algorithm iteratively expands (when computing least fixpoints) and contracts (when computing greatest fixpoints) an approximating automaton until the winning region is precisely recognised. Our method is a generalisation of Cachat's for solving Büchi games [13], which is itself a generalisation of the saturation technique for reachability analysis. However, we adopt a different proof strategy which we believe to be cleaner than Cachat's original proof. Our contribution can equivalently be presented as a solution to the *global* model checking problem: given a pushdown system $\mathcal{K}$, a modal mu-calculus formula $\chi(\overline{Y})$, and a regular valuation V, our method can *directly* compute an automaton that recognises the set $[\![\chi(\overline{Y})]\!]_V^{\mathcal{K}}$ of $\mathcal{K}$-configurations satisfying $\chi(\overline{Y})$ with respect to V.

Our algorithm has several advantages:

(i) The algorithm is simple and direct. Even though pushdown graphs are in general infinite, our construction of the automaton that recognises the winning region follows, in outline, the standard pen-and-paper calculation of the semantics of modal mu-calculus formulas in a *finite* transition system. Through the use of *projection*, our algorithm is guaranteed to terminate in a finite number of steps, even though the usual fixpoint calculations may require transfinite iterations. Thanks to projection, the state-sets of the approximating automata are bounded: during expansion, the number of transitions increases, but only up to the bound determined by the finite state-set; during contraction, the number of transitions decreases until it reaches zero or stabilises.

(ii) The correctness proof is simple and easy to understand. A conceptual innovation of the correctness argument are *valuation soundness* and *valuation completeness*. They are respectively under- and over-approximation conditions that apply *locally* to individual transitions of the automaton, rather than *globally* to the extensional

behaviour of the automaton (such as runs). By combining these conditions, which reduce the overhead of the proof, we show that our algorithm is both sound and complete in the usual sense.

(iii) Finally, our decision procedure builds on and extends the well-known saturation method, which is the implementation technique of choice of pushdown checkers. In contrast to previous solutions, our algorithm permits a straightforward optimisation that avoids an immediate exponential explosion, which we believe is important for an efficient implementation. Another advantage worth noting is that the automaton representing the winning region is independent of the maximum priority m (even though it takes time exponential in m to construct).

2 Preliminaries

A ***pushdown parity game*** is a parity game defined over a *pushdown graph* (i.e. the configuration graph of a pushdown system). Formally it is a quadruple $(\mathcal{P}, \mathcal{D}, \Sigma_\perp, \Omega)$ where $\mathcal{P} = \mathcal{P}_A \uplus \mathcal{P}_E = \{p^1, \ldots, p^z\}$ is a set of control states partitioned into Abelard's and Éloïse's states, $\Sigma_\perp := \Sigma \cup \{\perp\}$ is a finite stack alphabet (we assume $\perp \notin \Sigma$), $\mathcal{D} \subseteq \mathcal{P} \times \Sigma_\perp \times \mathcal{P} \times \Sigma_\perp^*$ is a set of pushdown rules and $\Omega : \mathcal{P} \to \{1, \ldots, m\}$ is a function assigning priorities to control states. As is standard, we assume that the bottom-of-stack symbol $\perp$ is neither pushed onto, nor popped from, the stack. We also assume there is a rule for each $p \in \mathcal{P}$ and $a \in \Sigma_\perp$.

A play begins from some configuration $\langle p, a\, w\rangle$. The player controlling p chooses $p\, a \to p'\, w' \in \mathcal{D}$ and the play moves to $\langle p', w'w\rangle$. Then, the player controlling p' chooses a move, and so on, generating an infinite run. The priority of a configuration $\langle p, w\rangle$ is $\Omega(p)$. A priority occurs infinitely often in a play if there are an infinite number of configurations with that priority. Éloïse wins the play if the smallest priority occurring infinitely often is even. Otherwise, Abelard is the winner.

A player's ***winning region*** of a pushdown parity game is the set of configurations from which the player can always win the game, regardless of the other player's strategy. Éloïse's winning region $\mathcal{W}_E$ of a parity game $\mathcal{G}$ is definable in the modal μ-calculus; the following is due to Walukiewicz [3]:

$$\mathcal{W}_E = [\![\mu Z_1.\nu Z_2. \ldots \mu Z_{m-1}.\nu Z_m.\varphi_E(Z_1, \ldots, Z_m)]\!]_V^{\mathcal{G}}$$

where m is the maximum parity (assumed even), V is a valuation of the variables[1], and

$$\varphi_E(Z_1, \ldots, Z_m) := \left(E \Rightarrow \bigwedge_{c \in \{1,\ldots,m\}} (c \Rightarrow \Diamond Z_c)\right) \wedge \left(\neg E \Rightarrow \bigwedge_{c \in \{1,\ldots,m\}} (c \Rightarrow \Box Z_c)\right)$$

where E is an atomic proposition asserting the current configuration is Éloïse's and, for $1 \leq c \leq m$, c asserts that the priority of the current control state is c.

For each $1 \leq c \leq m$, we have a variable Z_c. The odd priorities are bound by μ operators which can be understood intuitively as "finite looping". Dually, even priorities

[1] The valuation is initially empty since the formula has no free variables.

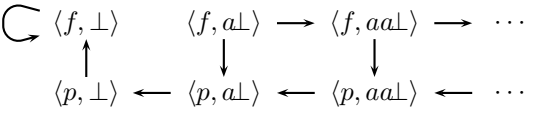
are bound by ν operators and can be understood as "infinite looping". The formula φ_E asserts that a variable Z_c is visited whenever a configuration of priority c is encountered. Thus the full formula asserts that the minimal priority occurring infinitely often must be even — otherwise a variable bound by the μ operator would be passed through infinitely often. It can be shown by a signature lemma that Éloïse has a winning strategy from a configuration satisfying the formula [3]. Since the formula's inverse is a similar formula with μ/ν, and $\Box/\Diamond$ reversed, Abelard has a winning strategy from any configuration not in $\mathcal{W}_E$.

Thanks to the Knaster-Tarski Fixpoint Theorem, the semantics of a fixpoint formula $[\![\sigma Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V$ where $\sigma \in \{\mu, \nu\}$ can be given as the limit of the sequence of α-***approximants*** $[\![\sigma^\alpha Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V$, where α ranges over the ordinals and λ ranges over the limit ordinals:

$$\begin{aligned}
[\![\sigma^0 Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V &:= \mathit{Init} \\
[\![\sigma^{\alpha+1} Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V &:= [\![\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_{V[Z \mapsto [\![\sigma^\alpha Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V]} \\
[\![\sigma^\lambda Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V &:= \bigcirc_{\alpha<\lambda} [\![\sigma^\alpha Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V
\end{aligned}$$

where $\mathit{Init} = \emptyset$ and $\bigcirc = \bigcup$ when $\sigma = \mu$, and Init is the set of all configurations and $\bigcirc = \bigcap$ when $\sigma = \nu$. The least ordinal κ such that $[\![\sigma^\kappa Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V = [\![\sigma Z.\chi(\overline{Y}, Z)]\!]^{\mathcal{G}}_V$ is called the *closure ordinal*.

Example 1. When interpreted in a pushdown graph, $\langle\, \sigma^\alpha Z.\chi(\overline{Y}, Z) \,\rangle_{\alpha \in \mathbf{Ord}}$ may have an infinite closure ordinal. Consider the following pushdown parity graph (which is a dual of an example of Cachat's [13]): all configurations are Abelard's, $\Omega(p) = 1$ and $\Omega(f) = 2$.



In this game, $\mathcal{W}_E = [\![\mu Z_1.\nu Z_2.\varphi_E(Z_1, Z_2)]\!]$ consists of all configurations. However, any $\langle f, a\, a^n \bot\rangle$ for some n only appears in an approximant of the least fixed point when $\langle f, a\, a\, a^n \bot\rangle$ and $\langle p, a\, a^n \bot\rangle$ appear in the previous approximant (since Abelard may move to either of these configurations). Hence, all $\langle p, a^n \bot\rangle$ must appear in the α-approximant before any $\langle f, a^n \bot\rangle$ can appear in the $(\alpha + 1)$-approximant. The first approximant containing all p configurations is the ω-approximant.

We use alternating multi-automata [1] as a representation of (regular) configuration-sets. Given a pushdown system $(\mathcal{P}, \mathcal{D}, \Sigma)$ with $\mathcal{P} = \{p^1, \ldots, p^z\}$, an ***alternating multi-automaton*** A is a tuple $(\mathcal{Q}, \Sigma, \Delta, I, \mathcal{F})$ where $\mathcal{Q}$ is a finite set of states, $\Delta \subseteq \mathcal{Q} \times (\Sigma \cup \{\, \bot \,\}) \times 2^{\mathcal{Q}}$ is a set of transitions (we assume $\bot \notin \Sigma$), $I = \{q^1, \ldots, q^z\} \subseteq \mathcal{Q}$ is a set of initial states, and $\mathcal{F} \subseteq \mathcal{Q}$ is a set of final states. Observe that there is an initial state for each control state of the pushdown system. We write $q \xrightarrow{a} Q$ just if $(q, a, Q) \in \Delta$; and define $q \xrightarrow{\varepsilon} \{q\}$; and $q \xrightarrow{aw} Q_1 \cup \cdots \cup Q_n$ just if $q \xrightarrow{a} \{q_1, \ldots, q_n\}$ and $q_k \xrightarrow{w} Q_k$ for all $1 \le k \le n$. Finally we define the *language accepted by* A, $\mathcal{L}(A)$, by: $\langle p^j, w\rangle \in \mathcal{L}(A)$ just if $q^j \xrightarrow{w} Q$ for some $Q \subseteq \mathcal{F}$. Henceforth, we shall refer to alternating multi-automata simply as ***automata***.

Reachability and Projection. The formula $\varphi_E(Z_1, \ldots, Z_m)$ asserts reachability in one step, which we compute using the reachability algorithm [1] due to Bouajjani *et al*. Cachat's extension of this algorithm requires a technique called *projection*. Using an example, we briefly introduce the relevant techniques.

Take a PDS with the rules $p^1\, a \to p^2\, \varepsilon$ and $p^2\, b \to p^2\, ba$. The automaton A_{eg} in Figure 1 represents a configuration set $\mathcal{C}$. Let $Pre(\mathcal{C})$ be the set of all configurations that can reach $\mathcal{C}$ in exactly one step. To calculate $Pre(\mathcal{C})$ we first add a new set of initial states — since we don't necessarily have $\mathcal{C} \subseteq Pre(\mathcal{C})$. By applying $p^1\, a \to p^2\, \varepsilon$, any configuration of the form $\langle p^1, aw\rangle$, where w is accepted from q^2 in A_{eg}, can reach $\mathcal{C}$. Hence we add an a-transition from q^1_{new}. (Via the pop transition, we reach $\langle p^2, w\rangle \in \mathcal{L}(A_{eg})$.) Alternatively, via $p^2\, b \to p^2\, ba$, any configuration of the form $\langle p^2, bw\rangle$, where baw is accepted from q^2 in A_{eg}, can reach $\mathcal{C}$. The push, when applied backwards, replaces ba by b. We add a b-transition from q^2_{new} which skips any run over ba from q^2. Figure 2 (i) shows the resulting automaton.

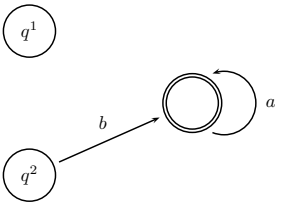


Fig. 1. The automaton A_{eg} accepting $\langle p^2, ba^*\rangle$

To ensure termination of the Büchi construction, Cachat uses *projection*, which replaces a new transition to an old initial state with a transition to the corresponding new state. Hence, the transition in Figure 2 (i) from q^1_{new} is *replaced* by the transition in Figure 2 (ii). The old initial states are then unreachable, and deleted, which, in this case, leaves an automaton with the same states as Figure 1 (modulo the new suffix) but an additional transition. In this sense, the state-set remains fixed.

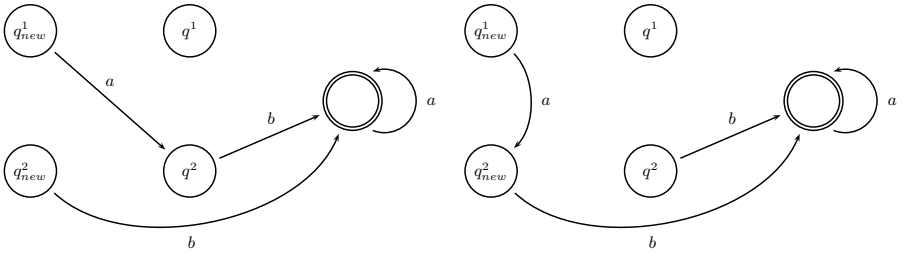


Fig. 2. (i) On the left, A_{eg} updated by the rules $p^1\, a \to p^2\, \varepsilon$ and $p^2\, a \to p^2\, ba$; and (ii) on the right, the result of projecting the automaton in (i)

3 An Example

We begin with an intuitive explanation of the algorithm by means of an example. Consider the pushdown game represented in Figure 3. Note that this diagram is a quotient of the infinite state space. Since the aim of this example is to give an overview of the flow of the algorithm, the behaviour of the pushdown system is kept simplistic. The subscripts indicate the priority of a configuration[2] and an arc labelled with $push_w$

[2] Our priorities here begin at 0. This does not change the algorithm significantly.

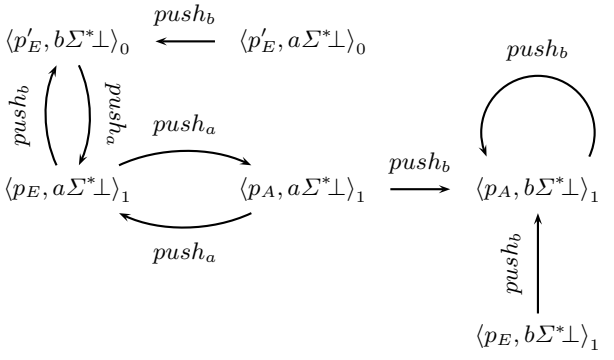


Fig. 3. An example pushdown parity game

indicates a pushdown rule of the form $p\,a \to p'\,w$ for some p, a and p'. Let $p_E, p'_E \in \mathcal{P}_E$ and $p_A \in \mathcal{P}_A$.

Éloïse can win from configurations of the forms $\langle p'_E, a\Sigma^*\bot\rangle_0, \langle p_E, a\Sigma^*\bot\rangle_1$, or $\langle p'_E, b\Sigma^*\bot\rangle_0$. Éloïse can loop between the last two of these configurations, generating a run with priority 0. From elsewhere, Abelard can force play to $\langle p_A, b\Sigma^*\bot\rangle_1$ and generate a run with priority 1. Computing Éloïse's winning region is equivalent to computing $[\![\nu Z_0.\mu Z_1.\varphi_E(Z_0, Z_1)]\!]^{\mathcal{G}}_V$. We illustrate how this is done in the following.

To compute a greatest fixed point, we begin by setting Z_0 to be the set of all configurations. We then calculate the automaton recognising the denotation of $\mu Z_1.\varphi_E(Z_0, Z_1)$ with this value of Z_0. The result is the value of Z_0 for the next iteration. After each iteration the value of Z_0 will be a subset of the previous value. This computation reaches a limit when the value of Z_0 stabilises, which is the denotation of the formula.

Computing the least fixed point proceeds in a similar manner, except that the initial value of Z_1 is set to $\emptyset$. We then compute the (automaton that recognises the) denotation of $\varphi_E(Z_0, Z_1)$, which gives us the next value of Z_1. Dual to the case of greatest fixed points, the value of Z_1 increases with each iteration.

Constructing the Automaton. (We shall often confuse the denotation of a formula with the automaton that recognises it, leaving it to the context to indicate which is intended.) We begin by setting Z_0 to the set of all configurations. The automaton recognising all configurations is shown in Figure 4 (i)[3]. Given this value of Z_0, we compute the denotation of $\mu Z_1.\varphi_E(Z_0, Z_1)$. The first step is to set the initial value of Z_1 to the empty set. The corresponding automaton is also shown in Figure 4 (ii). Observe that we have a separate set of initial states for Z_0 and Z_1.

We now compute $\varphi_E(Z_0, Z_1)$ which will be the next value of Z_1. A configuration $\langle p^j, aw\rangle$ with priority c should be accepted if Éloïse can play - or Abelard must play - a move which reaches some $\langle p^k, w'w\rangle \in V(Z_c)$. The result is Figure 4 (iii).

Observe that the computation of the new automaton has only added transitions. When computing a least fixed point, each generation of initial states has more transitions than the previous generation. In this example the number of possible transitions is finite since

[3] This is a simplification of the automaton defined in Section 4.

all transitions happen to go to q_f^*. Therefore, the automaton must eventually become saturated, causing termination. In the full algorithm, transitions from the new set of initial states to the old are *projected* back onto the new initial states. This ensures that the previous generation is not reachable. Hence, the state-set is fixed. When computing a greatest fixed point, termination can be proved dually: we begin with all transitions and iteratively remove transitions at each stage.

We now compute the next iterate of Z_1. We add a new set of initial states, and perform the reachability analysis, as in Figure 5 (i). If we were to perform another round of the reachability analysis, we would find a fixed point. That is, the transitions from the new initial states corresponding to Z_1 have the same outgoing transitions as the old. This fixed point is the next value of Z_0. Therefore, we set the current initial states of Z_1 to be the new initial states of Z_0. If necessary, we would also perform projections from the old initial states of Z_0 to the new. We then begin evaluating $\mu Z_1.\varphi_E(Z_0, Z_1)$ with our new value of Z_0. The initial value of Z_1 is the empty set, so we introduce new initial states corresponding to Z_1 with no outgoing transitions. Figure 5 (ii) shows the automaton after these steps.

We compute the next iterate of Z_1 as before, as in Figure 6. The second automaton is the fixed point of Z_1, and hence the new iterate of Z_0. Since the new Z_0 is identical to the previous Z_0, we have reached a final fixed point. Setting the initial states of Z_1 to be the initial states of Z_0, and deleting any unreachable states, gives the automaton in Figure 7, which accepts Éloïse's winning region.

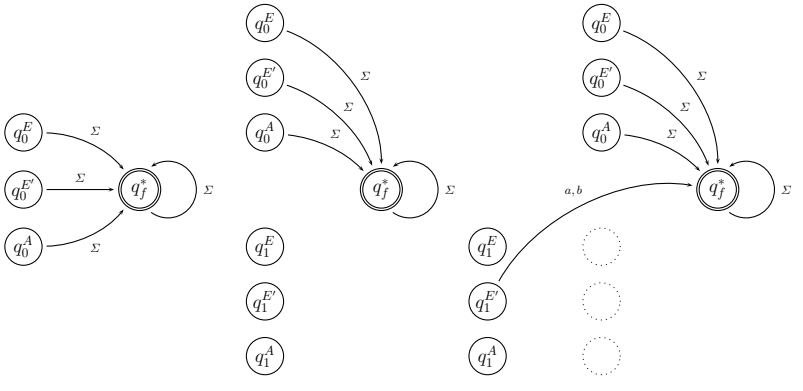


Fig. 4. From left to right, (i) the automaton accepting the initial value of Z_0; (ii) the automaton accepting the initial values of Z_0 and Z_1; and (iii) the automaton after the first round of reachability analysis

4 The Algorithm

Fix a pushdown parity game $\mathcal{G} = (\mathcal{P}, \mathcal{D}, \Sigma, \Omega)$ that has maximum priority m. The algorithm has two key components. The first — $Phi(A)$ — computes an automaton recognising $[\![\varphi_E(Z_1, \ldots, Z_m)]\!]_V^{\mathcal{G}}$, given an automaton A recognising the configuration-sets $V(Z_1), \ldots, V(Z_m)$. The second — $Sig(l, A)$ — computes, for each $1 \leq l \leq m$, an automaton recognising $[\![\sigma Z_l.\chi_{l+1}(Z_1, \ldots, Z_l)]\!]_V^{\mathcal{G}}$ where σ is either μ or ν as appropriate,

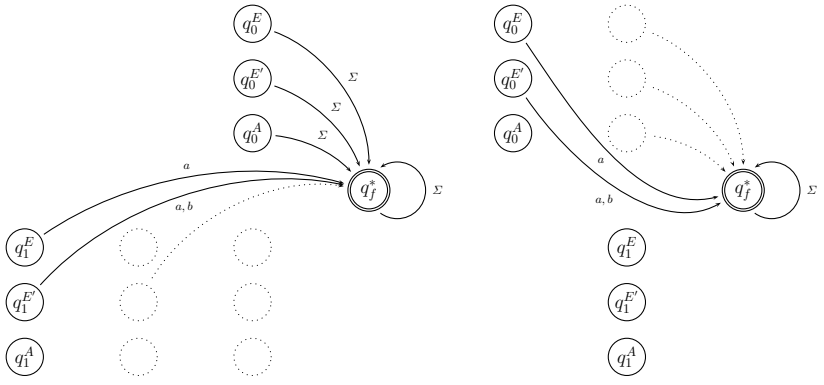


Fig. 5. (i) The automaton after the second round of reachability analysis; and (ii) the automaton with the new value of Z_0 and Z_1 set to the empty set

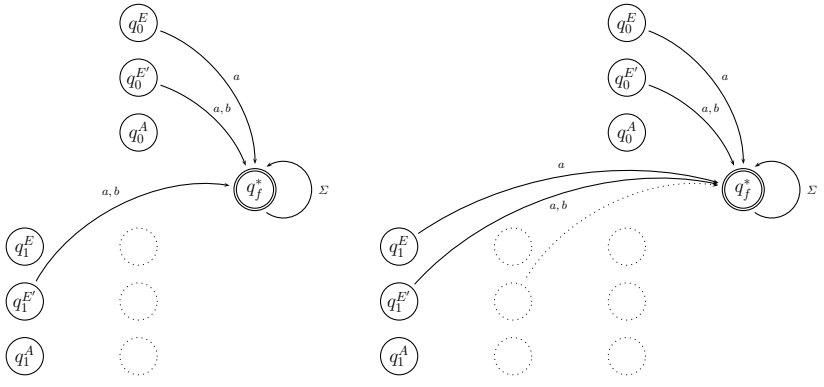


Fig. 6. The automaton after the first round of reachability analysis with the new Z_0; and the automaton after the second round of reachability analysis with the new Z_0

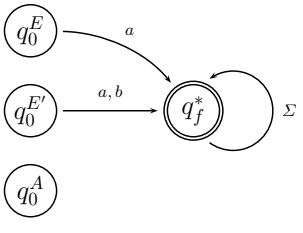


Fig. 7. The automaton accepting the winning region of Éloïse

given an automaton A recognising the configuration-sets $V(Z_1), \ldots, V(Z_{l-1})$, and

$$\chi_{l+1}(Z_1, \ldots, Z_l) := \sigma Z_{l+1} \ldots \sigma Z_m . \varphi_E(Z_1, \ldots, Z_m).$$

Format of the Automata. We describe the format of the automata constructed by the algorithm. Let $\mathcal{Q}_{all} := \{q^*, q_f^\varepsilon\}$, and $\mathcal{Q}_c := \left\{ q_c^j \mid 1 \leq j \leq |\mathcal{P}| \right\}$ for each $1 \leq c \leq m+1$. These states are used to give the valuations of the variables $Z_1, \ldots, Z_m$, and the semantics of $\varphi_E(Z_1, \ldots, Z_m)$ when $c = m+1$.

Let $0 \leq l \leq m+1$. An automaton A is said to be ***type-l*** just if:

(i) the state-set $\mathcal{Q}_A := \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_l \cup \mathcal{Q}_{all}$
(ii) every transition of the form $q_c^j \xrightarrow{a} Q$ has the property that $Q \neq \emptyset$, and for all j' and $c' > c$, $q_{c'}^{j'} \notin Q$ (i.e. there are no transitions to states with a higher priority)
(iii) the only final state is q_f^ε, which can only be reached by a $\bot$-transition (i.e. for each $q \xrightarrow{a} Q$, we have $q_f^\varepsilon \in Q$ iff $Q = \{q_f^\varepsilon\}$ iff $a = \bot$); further, q_f^ε has no outgoing transitions
(iv) we have $q^* \xrightarrow{\Sigma} \{q^*\}$ and $q^* \xrightarrow{\bot} \{q_f^\varepsilon\}$, and q^* has no other outgoing transitions.

It follows that there is a unique automaton of type-0.

In the following, let A be a type-l automaton, where $1 \leq c \leq l \leq m+1$. We define $\mathcal{L}_c(A) \subseteq \mathcal{P}\,\Sigma^*\,\bot$ by: for $1 \leq j \leq |\mathcal{P}|$, $\langle p^j, w\rangle \in \mathcal{L}_c(A)$ just if w is accepted by A from the initial state q_c^j. Thus $\mathcal{L}_c(A)$ is intended to represent the current valuation of the variable Z_c; in case $l = m+1$, $\mathcal{L}_{m+1}(A)$ is intended to represent $[\![\varphi_E(Z_1, \ldots, Z_m)]\!]_V^{\mathcal{G}}$ where the valuation V maps Z_c to $\mathcal{L}_c(A)$. If we omit the subscript and write $\mathcal{L}(A)$, we mean $\mathcal{L}_l(A)$. By abuse of notation, we define $\mathcal{L}_q(A) \subseteq \Sigma^*\bot \cup \{\,\epsilon\,\}$ to be the set of words accepted by A from the state q (note that $\mathcal{L}_{q^*}(A) = \Sigma^*\bot$ and $\mathcal{L}_{q_f^\varepsilon}(A) = \{\,\varepsilon\,\}$).

Definition of the Algorithm. Given a pushdown parity game $\mathcal{G}$, the algorithm presented in Figure 8 computes $\mathcal{W}_E$, the winning region of $\mathcal{G}$:

$$\mathcal{W}_E = [\![\mu Z_1.\nu Z_2.\ldots.\sigma Z_{m-1}.\sigma Z_m.\varphi_E(Z_1, \ldots, Z_m)]\!]_\emptyset^{\mathcal{G}}.$$

When computing $[\![\varphi_E(Z_1, \ldots, Z_m)]\!]_V^{\mathcal{G}}$ we may add an exponential number of transitions. To compute $[\![\sigma Z_l.\cdots.\sigma Z_m\,.\,\varphi_E(Z_1, \cdots, Z_m)]\!]_V^{\mathcal{G}}$ we may require an exponential number of iterations. Hence, in the worst case, the algorithm is (singly) exponential in the number of control states and the maximum priority m.

Theorem 1. *Given a pushdown parity game $\mathcal{G} = (\mathcal{P}, \mathcal{D}, \Sigma, \Omega)$, we can construct an automaton recognising the winning region of Éloïse in EXPTIME in $|\mathcal{P}| \cdot m$ where m is the maximum priority.*

The alternating multi-automaton returned by the algorithm, $Sig(1, A_0)$, has $n = |\mathcal{P}|+2$ states. The number of transitions is bounded by $n \cdot |\Sigma| \cdot 2^n$, which is independent of m.

5 Termination and Correctness

Termination. First an auxiliary notion of monotonicity for automaton constructions. Let $1 \leq l,\ l' \leq m+1$, and A and A' be type-l automata. We write $A \preceq A'$ to mean: for all q, a and Q, if $q \xrightarrow{a} Q$ is an A-transition then it is an A'-transition. We

Input: A pushdown parity game $\mathcal{G} = (\mathcal{P}, \mathcal{D}, \Sigma, \Omega)$ with maximum priority m.
Output: A type-1 automaton recognising $[\![\chi_1]\!]^{\mathcal{G}}$, the winning region of $\mathcal{G}$.

begin
  return $Sig(1, A_0)$ % A_0 is the unique type-0 automaton.
end

procedure $Sig(l, A)$

Input: $1 \leq l \leq m+1$;
  a type-$(l-1)$ automaton A as valuation of $Z_1, \cdots, Z_{l-1}$.
Output: A type-l automaton denoting $\sigma Z_l \cdots \sigma Z_m \,.\, \varphi_E(\overline{Z})$, relative to A.

1. if $l = m+1$ then return $Phi(A)$
2. $A^0 := \begin{cases} A \text{ with new states } \mathcal{Q}_l, \text{ but no new transitions} & \text{if } \sigma Z_l = \mu Z_l \\ A \text{ with new states } \mathcal{Q}_l, \text{ and all outgoing transitions obeying the format of the automata.} & \text{if } \sigma Z_l = \nu Z_l \end{cases}$
3. for $i = 0$ to ∞ do
4. $B^i := Sig(l+1, A^i)$
5. $A^{i+1} := Proj(l, B^i)$
6. if $A^i = A^{i+1}$ then return A^i

procedure $Phi(A)$

Input: A type-m automaton A as valuation of $\overline{Z} = Z_1, \cdots, Z_m$.
Output: A type-$(m+1)$ automaton denoting $\varphi_E(\overline{Z})$, relative to A.

1. (*1-Step Reachability*) Construct the automaton A' by adding new states $\{q^1_{m+1}, \ldots, q^{|\mathcal{P}|}_{m+1}\}$ and the following transitions to A. For each $1 \leq j \leq |\mathcal{P}|$, set $c := \Omega(p^j)$, and
   - if $p^j \in \mathcal{P}_E$ then $q^j_{m+1} \xrightarrow{a} Q$ if $q^k_c \xrightarrow{w} Q$ and $(p^k, w) \in Next(p^j, a)$
   - if $p^j \in \mathcal{P}_A$ then $q^j_{m+1} \xrightarrow{a} Q_1 \cup \cdots \cup Q_n$ if $q^{k_1}_c \xrightarrow{w_1} Q_1,\ \ldots,\ q^{k_n}_c \xrightarrow{w_n} Q_n$, and $Next(p^j, a) = \{(p^{k_1}, w_1), \ldots, (p^{k_n}, w_n)\}$

   where $Next(p^j, a) := \{\ (p^k, w) \mid p^j\, a \to p^k\, w\ \in\ \mathcal{D}\ \}$.
2. return A'.

procedure $Proj(l, A)$

Input: $1 \leq l \leq m$; a type-$(l+1)$ automaton A.
Output: A type-l automaton.

1. For each j, replace each transition $q^j_{l+1} \xrightarrow{a} Q$ with $q^j_{l+1} \xrightarrow{a} \pi^l(Q)$ where $\pi^l(Q) := \{q^{j'}_{l+1} \mid q^{j'}_l \in Q\} \cup (Q - \mathcal{Q}_l)$.
2. For each j, remove the state q^j_l.
3. For each j, rename the state q^j_{l+1} to q^j_l.

Fig. 8. Algorithm for computing winning region of a pushdown parity game

consider automaton constructions $\mathcal{T}$ (such as Sig, Phi and $Proj$) that transform type-l automata to type-l' automata. We say that $\mathcal{T}$ is *monotone* just if $\mathcal{T}(A) \preceq \mathcal{T}(A')$ whenever $A \preceq A'$.

To show that our winning-region construction procedure terminates, it suffices to prove the following.

Theorem 2 (Termination). *For every $1 \leq l \leq m+1$ and every type-$(l-1)$ automaton A, the procedure $Sig(l, A)$ terminates.*

We prove the theorem by induction on l. It is straightforward to establish the base case of $l = m+1$: $Phi(A)$ (where A is type-m) terminates. For the inductive case of $Sig(l, -)$ where $1 \leq l \leq m$, since $Sig(l+1, -)$ terminates by the induction hypothesis, and $Proj(l, -)$ clearly terminates, it remains to check that in the computation of $Sig(l, A)$ where A is type-$(l-1)$, there exists an $i \geq 0$ such that $A^i = A^{i+1}$. Since all automata of the same type have the same finite state-set (and $A^0, A^1, \ldots$ are all type-l) , it suffices to show (i) of the following Lemma.

Lemma 1 (Monotonicity). *We have the following properties.*

(i) Let $1 \leq l \leq m$ and A be a type-$(l-1)$ automaton. In $Sig(l, A)$:
  a. if $\sigma Z_l = \mu Z_l$ then $A^i \preceq A^{i+1}$ for all $i \geq 0$
  b. if $\sigma Z_l = \nu Z_l$ then $A^{i+1} \preceq A^i$ for all $i \geq 0$.
(ii) For every $1 \leq l \leq m+1$, the construction $Sig(l, -)$ is monotone.
(iii) For every $1 \leq l \leq m$, the construction $Proj(l, -)$ is monotone.

Correctness. To prove correctness, we introduce the notions of *valuation soundness* and *completeness*. Fix a pushdown parity game $\mathcal{G} = (\mathcal{P}, \mathcal{D}, \Sigma, \Omega)$. A *valuation profile* is a vector $\overline{S} = (S_1, \ldots, S_l)$ of configuration-sets (i.e. vertex-sets of the underlying configuration graph). We define the valuation $V_{\overline{S}} : Z_c \mapsto S_c$ induced by $\overline{S}$, which we extend to a map $V_{\overline{S}} : \mathcal{Q}_A \longrightarrow 2^{\Sigma^* \perp}$ on the states of a type-l automaton as follows:

$$V_{\overline{S}} := \begin{cases} q_c^j \mapsto \{\, w \mid \langle p^j, w \rangle \in S_c \,\} & 1 \leq j \leq |\mathcal{P}|,\ 1 \leq c \leq l \\ q^* \mapsto \Sigma^* \perp \\ q_f^\varepsilon \mapsto \{\, \varepsilon \,\} \end{cases}$$

Definition 1. Given a valuation profile $\overline{S}$ of length l, we say that a type-l automaton A is $\overline{S}$***-sound*** just if, for all q, a and w, if A has a transition $q \xrightarrow{a} Q$ such that $w \in V_{\overline{S}}(q')$ for all $q' \in Q$, then $a\, w \in V_{\overline{S}}(q)$.

By induction on the length of the word, valuation soundness extends to runs of a multi-automaton. We then obtain that all accepting runs are sound.

Lemma 2. *Let A be a $\overline{S}$-sound automaton.*

(i) For all q, w and w', if A has a run $q \xrightarrow{w} Q$ such that $w' \in V_{\overline{S}}(q')$ for all $q' \in Q$, then $w\, w' \in V_{\overline{S}}(q)$.
(ii) For all $q \in \mathcal{Q}_A$, $\mathcal{L}_q(A) \subseteq V_{\overline{S}}(q)$.

Proof

(i) We prove by induction on the length of the word w. When $w = a$, the property is just $\overline{S}$-soundness. Take $w = au$ and some run $q \xrightarrow{a} Q \xrightarrow{u} Q'$ such that for all $q' \in Q'$, we have $w \in V_{\overline{S}}(q')$. By the induction hypothesis, we have the property for the run $Q \xrightarrow{u} Q'$. Hence, we have for all $q' \in Q$ that, $uw' \in V_{\overline{S}}(q')$. Thus, from $\overline{S}$-soundness, we have $auw' \in V_{\overline{S}}(q)$.

(ii) Take an accepting run $q \xrightarrow{w} Q_f$ of A. We have for all $q' \in Q_f = \{q_f^{\varepsilon}\}, \varepsilon \in V_{\overline{S}}(q')$. Thanks to (i), we have $w \in V_{\overline{S}}(q)$. □

Definition 2. Given a valuation profile $\overline{S}$ of length l, we say that a type-l automaton A is $\overline{S}$***-complete*** just if, for all q, a and w, if $a\,w \in V_{\overline{S}}(q)$ then A has a transition $q \xrightarrow{a} Q$ such that $w \in V_{\overline{S}}(q')$ for all $q' \in Q$.

By induction on the length of the word, valuation completeness extends to runs. Furthermore, an accepting run always exists when required.

Lemma 3. *Let A be an $\overline{S}$-complete automaton.*

(i) For all q, w and w', if $w\,w' \in V_{\overline{S}}(q)$ then A has a run $q \xrightarrow{w} Q$ such that $w' \in V_{\overline{S}}(q')$ for all $q' \in Q$.
(ii) For all $q \in \mathcal{Q}_A$, $V_{\overline{S}}(q) \subseteq \mathcal{L}_q(A)$.

Notation. Recall $\chi_l(Z_1, \ldots, Z_{l-1}) := \sigma Z_l \cdots Z_m.\varphi_E(Z_1, \ldots, Z_m)$ where $1 \leq l \leq m+1$. Thus we have $\chi_1 = \mu Z_1 \ldots \sigma Z_m.\varphi_E(\overline{Z})$ and $\chi_{m+1}(Z_1, \ldots, Z_m) = \varphi_E(\overline{Z})$. Let $\overline{S} = (S_1, \ldots, S_{l-1})$; we write $(\overline{S}, T)$ to mean $(S_1, \ldots, S_{l-1}, T)$. Thus we write $\chi_l(\overline{S})$ to mean $\chi_l(S_1, \cdots, S_{l-1})$, and $\chi_{l+1}(\overline{S}, Z_l)$ to mean $\chi_{l+1}(S_1, \ldots, S_{l-1}, Z_l)$.

Proposition 1 (Main). *Let $1 \leq l \leq m+1$, A be a type-$(l-1)$ automaton, and $\overline{S}$ be a valuation profile of length $l-1$.*

*(i) (**Soundness Preservation**) If A is $\overline{S}$-sound, then $Sig(l, A)$ is a type-l automaton which is $(\overline{S}, [\![\chi_l(\overline{S})]\!])$-sound.*[4]
*(ii) (**Completeness Preservation**) If A is $\overline{S}$-complete, then $Sig(l, A)$ is a type-l automaton which is $(\overline{S}, [\![\chi_l(\overline{S})]\!])$-complete.*

Since the type-0 automaton A_0 is trivially sound and complete with respect to the empty valuation profile, we obtain the following as an immediate corollary.

Theorem 3 (Correctness). *The procedure call $Sig(1, A_0)$ terminates and returns a type-1 automaton which is $([\![\chi_1]\!])$-sound and $([\![\chi_1]\!])$-complete. Hence, thanks to Lemmas 2 and 3, for each $1 \leq j \leq |\mathcal{P}|$, $V_{[\![\chi_1]\!]}(q_1^j) = \mathcal{L}_{q_1^j}(Sig(1, A_0))$ i.e. the automaton $Sig(1, A_0)$ recognises the configuration set $[\![\chi_1]\!]$, which is the winning region of the pushdown parity game $\mathcal{G}$.*

[4] By $[\![\chi_l(S_1, \cdots, S_{l-1})]\!]$ we mean $[\![\chi_l(Z_1, \cdots, Z_{l-1})]\!]_V$ where V maps Z_c to S_c.

Proof of the Main Proposition. We prove Proposition 1 by induction on l. First the base case: $l = m + 1$.

Lemma 4. *Let $\overline{S}$ be a valuation profile of length m, and A a type-m automaton.*

(i) $Phi(A)$ is a type-$(m+1)$ automaton.
(ii) If A is $\overline{S}$-sound then $Phi(A)$ is $(\overline{S}, [\![\varphi_E(\overline{S})]\!])$-sound.
(iii) If A is $\overline{S}$-complete then $Phi(A)$ is $(\overline{S}, [\![\varphi_E(\overline{S})]\!])$-complete.

Proof

(i) We omit the straightforward proof.
(ii) Let $\overline{S'} = (\overline{S}, [\![\varphi_E(\overline{S})]\!])$ and $\Omega(p^j) = c$. Take any transition $q^j_{m+1} \xrightarrow{a} Q$ in $Phi(A)$ and stack w such that for all $q^{j'}_{c'} \in Q$, $\langle p^{j'}, w\rangle \in V_{\overline{S'}}(Z_{c'})$. For an Éloïse position, we abuse notation by interpreting $Next(p^j, a)$ as the singleton set containing the rule that led to the introduction of the new transition. Essentially, we present the proof for an Abelard position, which can be easily applied to Éloïse's positions. Since A is $\overline{S}$-sound and for all $(p^k, w_k) \in Next(p^j, a)$ we have $q^k_c \xrightarrow{w_k} Q_k \subseteq Q$, we know that $\langle p^k, w_k w\rangle \in V_{\overline{S'}}(Z_c)$. Hence all $\langle p^k, w_k w\rangle$ are in $V_{\overline{S'}}(Z_c)$, and $\langle p^j, aw\rangle \in V_{\overline{S'}}(Z_{m+1}) = [\![\varphi_E(\overline{Z})]\!]^{\mathcal{G}}_{V_{\overline{S'}}}$, since all moves, in the case of Abelard, and a move in the case of Éloïse, reach configurations in Z_c.
(iii) Take any configuration $\langle p^j, aw\rangle \in V_{\overline{S'}}(Z_{m+1}) = [\![\varphi_E(\overline{Z})]\!]^{\mathcal{G}}_{V_{\overline{S'}}}$. Let $\Omega(p^j) = c$. There exists an appropriate assignment $\{(p^{k_1}, w_1), \ldots, (p^{k_n}, w_n)\}$ to $Next(p^j, a)$ (as before) such that $\langle p^{k_h}, w_h w\rangle \in V_{\overline{S'}}(Z_c)$ for all $h \in \{1, \ldots, n\}$. Since A is assumed to be $\overline{S}$-complete, it follows that all $\langle p^{k_h}, w_h w\rangle$ have a complete run. In particular, we have a complete run $q^{k_h}_c \xrightarrow{w_h} Q_h$ for all h. Hence, by the definition of $Phi(A)$, there exists a transition $p^j \xrightarrow{a} Q$ that is complete. □

For the inductive case of $1 \leq l \leq m$, we present the proof when $\sigma Z_l = \mu Z_l$. The case of $\sigma Z_l = \nu Z_l$ is exactly dual (in outline, the soundness and completeness proofs are interchanged). Recall that $\chi_l(Z_1, \ldots, Z_{l-1}) := \sigma Z_l . \chi_{l+1}(Z_1, \cdots, Z_l)$.

Lemma 5. *Suppose $\sigma Z_l = \mu Z_l$. Let $\overline{S}$ be a valuation profile of length $l-1$, and A be a type-$(l-1)$ automaton; set $\theta = [\![\mu Z_l . \chi_{l+1}(\overline{S}, Z_l)]\!]$.*

(i) $Sig(l, A)$ is a type-l automaton.
(ii) If A is $\overline{S}$-sound, then $Sig(l, A)$ is $(\overline{S}, \theta)$-sound.
(iii) If A is $\overline{S}$-complete, then $Sig(l, A)$ is $(\overline{S}, \theta)$-complete.

Proof

(i) The result of the recursive call to $Sig(l+1, A)$ combined with the call to $Proj$ ensures the property.
(ii) Let $\overline{S'} := (\overline{S}, \theta)$. It is straightforward to see that A^0 is $\overline{S'}$-sound, since it did not add any transitions to A, which is assumed to be $\overline{S}$-sound. Hence, we assume by induction A^i is $\overline{S'}$-sound and argue the case for A^{i+1}.
Take a transition $q^j_l \xrightarrow{a} Q$ in A^{i+1} such that for all $q^k_{l'} \in Q$ we have $\langle p^k, w\rangle \in V_{\overline{S'}}(Z_{l'})$. Take the corresponding transition $q^j_{l+1} \xrightarrow{a} Q'$ in $Sig(l+1, A^i)$ before

the projection. In particular, for every $q_l^k \in Q$ we have q_l^k or q_{l+1}^k in Q'. By the induction hypothesis, we know $Sig(l+1, A^i)$ is $(\overline{S'}, [\![\chi_{l+1}(\overline{S'})]\!])$-sound. Furthermore, $V_{\overline{S'}}(Z_l) = \theta = [\![\chi_{l+1}(\overline{S}, \theta)]\!] = V_{\overline{S'}}(Z_{l+1})$. Since $Sig(l+1, A^i)$ is $(\overline{S'}, [\![\chi_{l+1}(\overline{S'})]\!])$-sound, we have $\langle p^j, aw\rangle \in V_{\overline{S'}}(Z_{l+1}) = V_{\overline{S'}}(Z_l)$ as required.

(iii) Let A be a type-$(l-1)$ automaton which is $\overline{S}$-complete. We use the shorthand $\theta^\alpha = [\![\mu^\alpha Z_l.\chi_{l+1}(\overline{S}, Z_l)]\!]$. We first show that if the type-l A^i is $(\overline{S}, \theta^\alpha)$-complete for some α then A^{i+1} is $(\overline{S}, \theta^{\alpha+1})$-complete. By the induction hypothesis, $B^i := Sig(l+1, A^i)$ is $(\overline{S}, \theta^\alpha, \theta^{\alpha+1})$-complete, since $\theta^{\alpha+1} = [\![\chi_{l+1}(\overline{S}, \theta^\alpha)]\!]$. We need to show that, after the projection, $A^{i+1} := Proj(l, B^i)$ is $\overline{S'}$-complete, where $\overline{S'} := (\overline{S}, \theta^{\alpha+1})$. Take some $\langle p^j, aw\rangle \in V_{\overline{S'}}(Z_l)$. We know B^i has a transition $q_{l+1}^j \xrightarrow{a} Q$ satisfying completeness. If Q contains no states of the form q_l^k, then the transition $q_l^j \xrightarrow{a} Q$ satisfies completeness in A^{i+1}. If Q contains states q_l^k, then $\langle p^k, w\rangle \in \theta^\alpha \subseteq \theta^{\alpha+1} = V_{\overline{S'}}(Z_l)$. Hence, we have a required complete transition after the projection, and so, A^{i+1} is $\overline{S'}$-complete. We require that $Sig(l, A)$ be $(\overline{S}, [\![\mu Z_l.\chi_{l+1}(\overline{S}, Z_l)]\!])$-complete. Take i such that $A^i = A^{i+1} = Sig(l, A)$. Trivially $Sig(l, A)$ is $(\overline{S}, \theta^0)$-complete. We proceed by transfinite induction. For a successor ordinal we know by induction that A^i is $(\overline{S}, \theta^\alpha)$-complete and from the above that A^{i+1} is $(\overline{S}, \theta^{\alpha+1})$-complete. Since $Sig(l, A) = A^i = A^{i+1}$ we are done. For a limit ordinal λ, we have that $Sig(l, A)$ is $(\overline{S}, \theta^\alpha)$-complete for all $\alpha < \lambda$. Since $\theta^\lambda = \bigcup_{\alpha<\lambda} \theta^\alpha$, the result follows because each configuration in the limit appears in some smaller approximant, and the transition witnessing completeness for the approximant witnesses completeness for the limit. □

6 Optimisation

In the procedure $Sig(l, A)$, in case $\sigma Z_l = \nu Z_l$, our definition of A^0 contains all allowable transitions, and hence is immediately exponential. However, if we have $q \xrightarrow{a} Q$ and $q \xrightarrow{a} Q'$ with $Q \subseteq Q'$, then acceptance from Q' implies acceptance from Q. That is, the transition to Q' is redundant. Furthermore, acceptance from any q_c^j implies acceptance from q^* (trivially). Using these observations, we can optimise our automaton. In the following definition, $Q \ll Q'$ can be taken to mean an accepting run from Q' implies an accepting run from Q.

Definition 3. *For all non-empty sets of states Q and Q', we define*

$$Q \ll Q' \;:=\; \big((q^* \in Q \Rightarrow \exists q.q \neq q_f^\varepsilon \wedge q \in Q') \;\wedge\; (\forall q \neq q^*.q \in Q \Rightarrow q \in Q')\big)$$

and $\textsc{Expand}(A) := \{\, q \xrightarrow{a} Q' \mid q \xrightarrow{a} Q \text{ in } A \text{ and } Q \ll Q' \,\}$.

By specifying monotonicity with respect to $\textsc{Expand}(A)$ rather than A, A^0 (in case $\sigma Z_l = \nu Z_l$) only needs transitions to q^* and q_f^ε, which is linear. When this optimisation is used in the case of a one-player game, the constructed automaton will not use any alternating transitions. Furthermore, we can remove redundant transitions at every stage of the algorithm. Since a transition to $\{q^*\}$ is powerful with respect to $\ll$ we expect to keep the automaton small. However, this will have to be confirmed experimentally.

To test termination of $Sig(A, l)$, we check if $\textsc{Expand}(A^{i+1}) = \textsc{Expand}(A^i)$.

Lemma 6. $\textsc{Expand}(A) \preceq \textsc{Expand}(A')$ *if and only if whenever* $q \xrightarrow{a} Q$ *in* A *then there is some* $Q' \ll Q$ *with* $q \xrightarrow{a} Q'$ *in* A'.

By induction, we extend the property to runs. Hence $\textsc{Expand}(A) \preceq \textsc{Expand}(A')$ implies $\mathcal{L}(A) \subseteq \mathcal{L}(A')$. Finally, we have:

Lemma 7. *The optimisation preserves monotonicity and both valuation soundness and valuation completeness.*

Conclusion. We have proposed a new, simple and direct algorithm for computing the winning region of a pushdown parity game. The algorithm uses a mu-calculus formula that characterises Éloïse's winning region as a guide to construct the required automaton. We have identified an optimisation that avoids an immediate exponential blow up. An interesting open problem is to construct winning strategies using our approach.

Acknowledgments. This work is supported by EPSRC (EP/F036361). We are greatly indebted to Arnaud Carayol for his invaluable assistance.

References

1. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
2. Finkel, A., Willems, B., Wolper, P.: A direct symbolic approach to model checking pushdown systems. In: INFINITY (1997)
3. 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)
4. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: FOCS 1991, pp. 368–377 (1991)
5. Esparza, J., Kučera, A., Schwoon, S.: Model-checking LTL with regular valuations for pushdown systems. In: Kobayashi, N., Pierce, B.C. (eds.) TACS 2001. LNCS, vol. 2215, pp. 316–339. Springer, Heidelberg (2001)
6. Hague, M.: Saturation methods for global model-checking pushdown systems. PhD. Thesis, University of Oxford (2009)
7. Jones, N., Muchnick, S.: Even simple programs are hard to analyse. JACM 24, 338–350 (1977)
8. Piterman, N., Y. Vardi, M.: Global model-checking of infinite-state systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 387–400. Springer, Heidelberg (2004)
9. Serre, O.: Note on winning positions on pushdown games with ω-regular conditions. Information Processing Letters 85, 285–291 (2003)
10. Schwoon, S.: Model-checking Pushdown Systems. PhD thesis, Tech. Univ., Munich (2002)
11. Ball, T., Rajamani, S.K.: Bebop: A Symbolic Model Checker for Boolean Programs. In: Havelund, K., Penix, J., Visser, W. (eds.) SPIN 2000. LNCS, vol. 1885, pp. 113–130. Springer, Heidelberg (2000)
12. Ball, T., Rajamani, S.K.: The SLAM project: Debugging system software via static analysis. In: POPL, pp. 1–3 (2002)
13. Cachat, T.: Games on Pushdown Graphs and Extensions. PhD thesis, RWTH Aachen (2003)
14. Reps, T., Schwoon, S., Jha, S., Melski, D.: Weighted pushdown systems and their application to interprocedural dataflow analysis. Sci. Comput. Program. (2005)