Uniform Solution of Parity Games on Prefix-Recognizable Graphs*

Thierry Cachat

Lehrstuhl für Informatik VII, RWTH, D-52056 Aachen
Fax: (49) 241-80-22215, Email: cachat@informatik.rwth-aachen.de

Abstract

Walukiewicz gave in 1996 a solution for parity games on pushdown graphs: he proved the existence of pushdown strategies and determined the winner with an EXPTIME procedure. We give a new presentation and a new algorithmic proof of these results, obtain a uniform solution for parity games (independent of their initial configuration), and extend the results to prefix-recognizable graphs. The winning regions of the players are proved to be effectively regular, and winning strategies are computed.

1 Introduction

Games are an important model of reactive computation and a versatile tool for the analysis of logics like the μ-calculus [4, 5]. Namely we know that the model checking problem of the μ-calculus is polynomialy equivalent to the problem of solving parity games. In recent years, games over infinite graphs have attracted attention as a framework for the verification and synthesis of infinite-state systems [6].

In the present paper we consider two-player parity games on pushdown graphs (transition graphs of pushdown automata) and on prefix-recognizable graphs. It was shown in [10] that one can determine in EXPTIME the winner of a pushdown game, and that winning strategies can be realized also by pushdown automata.

The drawback of these results [6, 10] is a dependency of the analysis on a given initial game position, and a lack of algorithmic description of the (computation of) winning strategies. In this paper we extend the results of [10] to a uniform solution for parity games on

*a preliminary version of this paper was accepted at the Workshop INFINITY 2002, Brno

prefix-recognizable graphs (independent of initial configuration), and we define explicitly the (computation of a) winning strategy.

In Section 2 we give a new presentation and proof of the results of [10] stressing upon effectivity. Section 3 presents an exemple of pushdown game. Then in Section 4 we extend these results to compute uniformly the winning region of the game (the set of configurations from which Player I can win). It is proved to be effectively regular, and a corresponding winning pushdown strategy is also uniformly defined. In Section 5 we consider parity games on prefix-recognizable graphs, which are an extension of pushdown graphs, where the degree of a vertex can be infinite [2]. We show that any prefix-recognizable game can be "simulated" by a pushdown game, in the sense that under a certain correspondence of game positions, the winner of one game is the same player as the winner of the other game. An exemple is also provided. Applying the uniform solution of Section 4, we get a uniform solution and an effective winning strategy also over prefix-recognizable graphs.

The result of Section 4 has been found independently from us by Olivier Serre in [7].

2 Pushdown Games: Walukiewicz's Results

Sections 3 and 4 of [10] are not stated in an effective (*i.e.*, algorithmic) framework, and their results "become" effective only with the help of Section 5 of [10]. We prefer to give first a new presentation of the construction of Section 5 of [10]. Then the most important results can be deduced, including all algorithmic claims.

The idea of [10] is to "reduce" the pushdown game to a parity game on a finite graph. This allows to determine the winner, and also the winning strategy. We assume the positional ("memoryless") determinacy of parity games over finite graphs, see [4].

A *Finite State Parity game* (FSP) is a tuple (S, E, λ) where $S = S_I \uplus S_{II}$ is the finite set of vertices of the game graph, $E \subseteq S \times S$ is the edge relation, and $\lambda : S \longrightarrow \{0, \cdots, max - 1\}$ is the priority function ($max > 0$). It is assumed in [10] that $E \subseteq (S_I \times S_{II}) \cup (S_{II} \times S_I)$, but this is not essential. From now on we use the infix notation $\rightarrow$ for the edge relation: $\forall s, s' \in S$, $(s, s') \in E \Leftrightarrow s \rightarrow s'$.

Starting in a given initial vertex $\pi_0 \in S$, a play in (S, E, λ) proceeds as follows: if $\pi_0 \in S_I$, Player I picks the first transition (move) to π_1, else Player II does, and so on from the new vertex π_1. A play is a (possibly infinite) maximal sequence $\pi_0\pi_1 \cdots$ of successive vertices. For the winning condition we consider the max-parity version: Player I wins the play $\pi_0\pi_1 \cdots$ iff $\limsup_{k\to\infty} \lambda(\pi_k)$ is odd, *i.e.*, iff the maximal priority seen infinitely often in the play is odd.

A *Pushdown Game System* (PDS) is a tuple (P, Γ, Δ), where $P = P_{I} \uplus P_{II}$ is the partitioned set of control locations, Γ the stack alphabet, and Δ a (finite) transition relation $\Delta \subseteq P \times \Gamma \times P \times \Gamma^{\leqslant 2}$, where $\Gamma^{\leqslant 2} = \epsilon \cup \Gamma \cup \Gamma^2$. The set of configurations of the PDS is $V = P\Gamma^*$, partitioned into $V_I = P_I\Gamma^*, V_{II} = P_{II}\Gamma^*$. The set of transitions, or edge relation, is $\{(p\gamma\nu, p'\mu\nu) \mid (p, \gamma, p', \mu) \in \Delta, \nu \in \Gamma^*\}$. We also have a priority function $\Omega : P \longrightarrow \{0, \cdots, max - 1\}$, extended to V by $\Omega(p\nu) = \Omega(p)$. A play starting from an initial configuration π_0, and the winning condition are defined in the same way as for the FSP, replacing S_I and S_{II} by V_I and V_{II}. Player I wins a play $\pi_0\pi_1 \cdots$ iff $\limsup_{k\to\infty} \Omega(\pi_k)$ is odd. In this section we consider a particular initial configuration $\pi_0 = p_0 \perp$, where $\perp \in \Gamma, p_0 \in P$.

A *pushdown strategy* for I in its general form is a deterministic pushdown automaton with input and output. It "reads" the moves of Player II (elements of Δ) and outputs the moves (choices) of Player I, like a pushdown transducer.

Definition 2.1 *Given a game over a PDS (P, Γ, Δ), where Δ_σ is the set of transition rules in Δ departing from Player σ configurations, a* pushdown strategy *for Player* I *in this game is a* deterministic *pushdown automaton $\mathcal{S} = (Q, A, \Pi)$, with a set Q of control states, some stack alphabet A, and a finite transition relation $\Pi \subseteq ((Q \times A \times \Delta_{II}) \times (Q \times A^*)) \cup ((Q \times A) \times (Q \times A^* \times \Delta_I))$.*

Theorem 2.2 [10] *Given a Pushdown Game System $\mathcal{G}$ with a parity winning condition, one can construct a Finite State Parity game such that:*
1. the winner of the parity game over $\mathcal{G}$ from the initial configuration $p_0 \perp$ is the winner of the FSP from a certain initial vertex denoted $\mathtt{Check}(p_0, \perp, B, \Omega(p_0))$, where $B \in (\mathcal{P}(P))^{max}$,
2. a winning pushdown strategy for Player I *in the parity game over $\mathcal{G}$ from the initial configuration $p_0 \perp$ can be constructed from a winning strategy for Player* I *in the FSP from the initial vertex mentioned above..*

In the sequel we present informally how a play on the PDS is "simulated" in the FSP. We will see later that a configuration $p\gamma\nu$ of the PDS, where $p \in P, \gamma \in \Gamma$, and $\nu \in \Gamma^*$, is represented in the FSP by a vertex $\mathtt{Check}(p, \gamma, B, m)$, where $m \in \{0, \cdots, max - 1\}$ is a priority, and $B \in (\mathcal{P}(P))^{max}$ "summarizes" information about ν. (the number m is the highest priority seen in a certain part of the game, and B represents the set of control states q such that Player I can win the game from $q\nu$, under certain conditions depending on m). To begin with, consider the initial configuration $(p_0 \perp)$, where the symbol $\perp \in \Gamma$ cannot be erased w.r.t. the rules of Δ. The corresponding vertex of the FSP is $\mathtt{Check}(p_0, \perp, B, m)$, where in this particular case B and m are not relevant.

From a configuration $p\gamma v$, simulated by $\mathtt{Check}(p,\gamma,B,m)$, the player whose turn is it is determined by p, in the PDS as well as in the FSP: either $p \in P_{\mathrm{I}}$ or $p \in P_{\mathrm{II}}$. Let $\sigma \in \{\mathrm{I}, \mathrm{II}\}$ such that $p \in P_\sigma$. Different types of moves are possible in the PDS.

If Player σ chooses a transition (p,γ,p',γ'), *i.e.*, if the stack length remains constant, then the FSP proceeds to the vertex $\mathtt{Check}(p',\gamma',B,\max(m,\Omega(p')))$. This means that B remains the same, m is updated for later use and represents the maximal priority seen since last initialization of m (see below). The priority of this vertex is $\Omega(p')$ in the FSP, as well as the corresponding configuration in the PDS, and the play goes on like that until some "push" or "pop" operation occurs.

The key point is the treatment of the push operation, because one cannot store in the FSP the whole information contained in the stack. If Player σ chooses a transition $(p,\gamma,p',\gamma'\eta) \in \Delta$, *i.e.*, "pushes" one more symbol onto the stack, then in the FSP the corresponding new vertex is $\mathtt{Push}(B,m,p',\gamma'\eta)$. This is an intermediate vertex were Player I (always) has to make a decision. He has to guess what can happen later, and what he can guarantee. Player I chooses a tuple $C \in (\mathcal{P}(P))^{max}$ such that he claims/guesses that whenever the symbol γ' currently at the top of the stack is "popped", then after this pop operation, the PDS will be in a control-state $q \in C_\ell$ such that ℓ is the highest priority seen between the "push" and the "pop" of this γ'. This part of the game is a "subgame" in [10], and this notion is not so far from the idea of "detour" in [8]. More precisely, γ' can be replaced later by another letter, but the condition on C must hold when the length of the stack decreases and symbol η comes at the top of the stack.

So Player I goes to the vertex $\mathtt{Claim}(B,m,p',\gamma'\eta,C)$, which is a vertex of Player II . In particular, if $C = (\emptyset,\cdots,\emptyset)$, then Player I is claiming that the stack will never become again shorter. And Player I can claim that the highest priority that can be seen in the subgame is ℓ by choosing C as $(C_1,\cdots,C_\ell,\emptyset,\cdots,\emptyset)$. Player II has to answer the claim of Player I : either he thinks that Player I is bluffing, and he challenges the claim, or he believes that Player I can achieve his claim, and he wants to see what happens after the subgame.

The second case is simple: Player II goes to vertex $\mathtt{Jump}(q,\eta,B,m,\ell)$ such that $q \in C_\ell$. This is an intermediate vertex which, as a shortcut, simulates one of the above mentioned subgames: among the propositions of Player I , Player II chooses that the highest priority seen in this subgame was ℓ, and when η appears again at the top of the stack, the new control state is q. The priority of this $\mathtt{Jump}()$ vertex is ℓ in the FSP. Then the play goes on to $\mathtt{Check}(q,\eta,B,\max(\ell,m,\Omega(q)))$ without any alternative.

In the first case, when Player II challenges the claim, he goes to vertex $\mathtt{Check}(p',\gamma',C,\Omega(p'))$. This means that the last component is reset to $\Omega(p')$, and will remember the maximal priority seen in the subgame we just entered. The tuple C is stored, and whenever a "pop" operation occurs later, it is possible to check if the claim of Player I is achieved. If it is, this means immediate win for Player I . If it is not, this means immediate win for Player II (see the proof for details, and above for the update of m). But the play can also stay forever in the Check() vertices, *i.e.*, without "pop". In this case the winner is determined by the parity condition. In fact the claim of Player I after a push operation means also that if no pop occurs later, then he has to win the subgame just with the parity condition.

We restrict ourselves in this paper to the following form of pushdown strategy. We consider a strategy automaton (Q, A, Π) where $Q = P = P_{\mathrm{I}} \uplus P_{\mathrm{II}}$, $A = \Gamma \times \Sigma$, Σ is any alphabet, and $\Pi \subseteq ((P_{\mathrm{II}} \times A \times \Delta_{\mathrm{II}}) \times (P \times A^*)) \cup ((P_{\mathrm{I}} \times A) \times (P \times A^* \times \Delta_{\mathrm{I}}))$. Moreover we have the condition that whenever the game is in a configuration $p\gamma_0 \cdots \gamma_n$, the strategy automaton should be in a configuration $p(\gamma_0, \sigma_0) \cdots (\gamma_n, \sigma_n)$, which means that the strategy has more information in its stack, represented by $\sigma_0 \cdots \sigma_n$, but follows the play. If $p \in P_{\mathrm{I}}$, then $p(\gamma_0, \sigma_0)$ determines the move of Player I w.r.t. Π, and the strategy updates its stack. If $p \in P_{\mathrm{II}}$, then for any move of Player II , *i.e.*, for any transition in Δ_{II}, the strategy should update its stack. At the beginning of the play, the strategy has to be initialized properly, according to the initial configuration of the game. Then for each move of the play, the strategy executes a transition.

In our particular form of strategy, there is a redundancy in the transition relation: suppose $p(\gamma_0, \sigma_0)$ is the top of the stack, if $p \in P_{\mathrm{I}}$, then a unique transition is possible in the strategy, and the output of the move $\delta_{\mathrm{I}} \in \Delta_{\mathrm{I}}$ can be deduced from the update of the stack. If $p \in P_{\mathrm{II}}$, then a unique transition can follow the choice of Player II and update the stack accordingly, so the input of $\delta_{\mathrm{II}} \in \Delta_{\mathrm{II}}$ is redundant. From now on we consider $\Pi \subseteq (P \times A) \times (P \times A^{\leqslant 2})$.
Formally, if $p \in P_{\mathrm{I}}$, then $\forall a \in A\ \exists!(p, a, p', w) \in \Pi$. Moreover if $(p, a, p', w) \in \Pi$ and $a = (\gamma, \sigma)$, $w = (\gamma_1, \sigma_1) \cdots (\gamma_k, \sigma_k)$ ($2 \geqslant k \geqslant 0$), then $(p, \gamma, p', \gamma_1 \cdots \gamma_k) \in \Delta$, that is to say the hint of the strategy is valid. If $p \in P_{\mathrm{II}}$, then $\forall (p, \gamma, p', \gamma_1 \cdots \gamma_k) \in \Delta\ \exists!(p, a, p', w) \in \Pi$ such that $a = (\gamma, \sigma)$ and $w = (\gamma_1, \sigma_1) \cdots (\gamma_k, \sigma_k)$ ($2 \geqslant k \geqslant 0$).

Proof of Theorem 2.2

Definition of the FSP

The PDS is given by $\mathcal{G} = (P, \Gamma, \Delta)$, $P = P_{\mathrm{I}} \uplus P_{\mathrm{II}}$, and Ω.
For every $p, p', q \in P; \gamma, \gamma', \eta \in \Gamma; m, \ell \in \{0, \cdots, max - 1\}; B, C \in (\mathcal{P}(P))^{max}$, the FSP has the following vertices:

$$\mathsf{Check}(p, \gamma, B, m), \quad \mathsf{Push}(B, m, p', \gamma'\eta),$$
$$\mathsf{Claim}(B, m, p', \gamma'\eta, C), \quad \mathsf{Jump}(p, \gamma, B, m, \ell), \quad \mathit{Win}_{\mathrm{I}}(p), \mathit{Win}_{\mathrm{II}}(p),$$

and the following transitions:

$\mathtt{Check}(p,\gamma,B,m) \to \mathtt{Check}(p',\gamma',B,\max(m,\Omega(p')))$ if $(p,\gamma,p',\gamma') \in \Delta$,

$\mathtt{Check}(p,\gamma,B,m) \to \mathtt{Win}_{\mathrm{I}}(p')$ if $(p,\gamma,p',\epsilon) \in \Delta$ and $p' \in B_m$,

$\mathtt{Check}(p,\gamma,B,m) \to \mathtt{Win}_{\mathrm{II}}(p')$ if $(p,\gamma,p',\epsilon) \in \Delta$ and $p' \notin B_m$,

$\mathtt{Check}(p,\gamma,B,m) \to \mathtt{Push}(B,m,p',\gamma'\eta)$ if $(p,\gamma,p',\gamma'\eta) \in \Delta$,

$\mathtt{Push}(B,m,p',\gamma'\eta) \to \mathtt{Claim}(B,m,p',\gamma'\eta,C)$,

$\mathtt{Claim}(B,m,p',\gamma'\eta,C) \to \mathtt{Check}(p',\gamma',C,\Omega(p'))$,

$\mathtt{Claim}(B,m,p',\gamma'\eta,C) \to \mathtt{Jump}(q,\eta,B,m,\ell)$ if $q \in C_\ell$, and

$\mathtt{Jump}(q,\eta,B,m,\ell) \to \mathtt{Check}(q,\eta,B,\max(\ell,m,\Omega(q)))$.

One defines in the FSP the player whose turn it is: $\mathtt{Check}(p,\gamma,B,m) \in S_{\mathrm{I}} \Leftrightarrow p \in P_{\mathrm{I}}$, but $\mathtt{Push}(B,m,p',\gamma'\eta) \in S_{\mathrm{I}}$ and $\mathtt{Claim}(B,m,p',\gamma'\eta,C) \in S_{\mathrm{II}}$. From other vertices, the players have no alternative: there is a unique successor.

One has the following priorities:

$\lambda(\mathtt{Check}(p,\gamma,B,m)) = \Omega(p)$, $\lambda(\mathtt{Jump}(q,\gamma,B,m,\ell)) = \ell$, and

$\lambda(\mathtt{Push}(B,m,p',\gamma'\eta)) = \lambda(\mathtt{Claim}(B,m,p',\gamma'\eta,C)) = 0$ because the latter are intermediate vertices which should not interfere with the parity condition.

It remains to clarify the situation concerning deadlocks. If the first letter of the stack and the control state do not permit to execute a transition, there is a deadlock in the PDS as in the corresponding vertex of the FSP. We leave to the reader to choose the convention concerning which player wins in that case.

If one needs a bottom stack symbol ($\perp$), that cannot be erased and cannot be pushed, one has to care for this explicitly in Γ and Δ. Otherwise when the stack is empty, no transition is possible in our framework of PDS. We have again to choose a convention for this type of deadlock. It concerns the choice of B in the initial vertex $\mathtt{Check}(p_0,\perp,B,\Omega(p_0))$ of the FSP.

Equivalence between the games: from FSP to PDS

Suppose that Player I has a winning strategy in the FSP from vertex $\mathtt{Check}(p_0,\perp,B,\Omega(p_0))$. Since the game graph is finite, and the strategy can be taken positional [4], it is effectively given as a subset of the set of transitions, and denoted $\overset{str}{\to} \subseteq \to$. We will define from it a winning pushdown strategy in the PDS. This construction will be effective.

The strategy automaton is (P,A,Π), with $A = \Gamma \times \Sigma$. We fix $\Sigma = (\mathcal{P}(P))^{max} \times \{0,\cdots,max-1\}$. For notational convenience, an element $(\gamma,(B,m))$ of A will be written γBm, and a transition $((p,\gamma Bm),(p',\gamma'B'm')) \in \Pi$ will be written as a prefix rewriting rule $p\,\gamma Bm \overset{str}{\hookrightarrow} p'\,\gamma'B'm'$. Similarly $p\,\gamma Bm \overset{str}{\hookrightarrow} p'\,\epsilon$, and $p\,\gamma Bm \overset{str}{\hookrightarrow} p'\,\gamma'B'm'\,\gamma''B''m''$.

The initial configuration of the PDS is $p_0 \perp$, and the one of the FSP is $\mathsf{Check}(p_0, \perp, B, \Omega(p_0))$, where B is chosen according to the convention about empty stack (see above). The initial configuration of the strategy is $p_0 \perp B\Omega(p_0)$. From a configuration $p\,\gamma B m\, w$ of the strategy automaton, where $w \in A^*$, the transition in Π is defined as follows:

If $p \in P_I$, then we know that in the FSP Player I chooses the next vertex from $\mathsf{Check}(p, \gamma, B, m)$ according to $\stackrel{str}{\rightarrow}$.

- If $\mathsf{Check}(p, \gamma, B, m) \stackrel{str}{\rightarrow} \mathsf{Check}(p', \gamma', B, \max(m, \Omega(p')))$, then use the transition $p\,\gamma B m \stackrel{str}{\hookrightarrow} p'\,\gamma' B \max(m, \Omega(p'))$.

- If $\mathsf{Check}(p, \gamma, B, m) \stackrel{str}{\rightarrow} \mathit{Win}_I(p')$, then apply $p\,\gamma B m \stackrel{str}{\hookrightarrow} p'\,\epsilon$. Of course in the PDS there is no immediate win, but the game goes on (cf jump move). Moreover it is necessary, in the new top letter $\gamma' B' m'$ of the stack to update m' according to m and $\Omega(p')$, as follows (details are left to the reader): $p\,\gamma B m \gamma' B' m' \stackrel{str}{\hookrightarrow} (p', m)\gamma' B' m' \stackrel{str}{\hookrightarrow} p'\,\gamma' B' \max(m, m', \Omega(p'))$.

- If $\mathsf{Check}(p, \gamma, B, m) \stackrel{str}{\rightarrow} \mathsf{Push}(B, m, p', \gamma'\eta) \stackrel{str}{\rightarrow} \mathsf{Claim}(B, m, p', \gamma'\eta, C)$, then apply $p\,\gamma B m \stackrel{str}{\hookrightarrow} p'\,\gamma' C\Omega(p')\,\eta B m$. Of course in the PDS Player II has no opportunity to jump, he must enter the subgame.

If $p \in P_{II}$, then Player II chooses any possible transition in the PDS, and the Strategy automaton updates its stack according to the winning strategy $\stackrel{str}{\rightarrow}$ of the FSP. More precisely, if Player II chooses $(p, \gamma, p', \gamma') \in \Delta$, then the strategy executes $p\,\gamma B m \stackrel{str}{\hookrightarrow} p'\,\gamma' B \max(m, \Omega(p'))$. If II chooses $(p, \gamma, p', \epsilon) \in \Delta$, then the strategy choses $p\,\gamma B m \stackrel{str}{\hookrightarrow} p'\,\epsilon$, followed by an update of m'. If II chooses $(p, \gamma, p', \gamma'\eta) \in \Delta$, then we have to follow $\stackrel{str}{\rightarrow}$ in the FSP, and find C such that $\mathsf{Push}(B, m, p', \gamma'\eta) \stackrel{str}{\rightarrow} \mathsf{Claim}(B, m, p', \gamma'\eta, C)$. Then $p\,\gamma B m \stackrel{str}{\hookrightarrow} p'\,\gamma' C\Omega(p')\,\eta B m$ is applied.

Because $\stackrel{str}{\rightarrow}$ is winning in the FSP, $\stackrel{str}{\hookrightarrow}$ is also winning in the PDS. Moreover using known algorithms to solve the FSP, we have constructed a pushdown strategy which is winning.

from PDS to FSP

Given a winning strategy in the PDS, we will define a winning strategy in the FSP. Here a strategy in the PDS from initial configuration $p_0 \perp = \pi_0$ is a function Str which associates to the prefix $\pi_0 \cdots \pi_n$ of a play a "next move", *i.e.*, a transition in Δ. We consider a strategy for Player I, so it is defined if $\pi_n \in V_I$. This function is not necessarily computable, so this part is not effective.

As above, a vertex $\mathsf{Check}(p, \gamma, B, m)$ corresponds to a configuration $p\gamma v$ of the PDS. If $p \in P_{II}$, the PDS has to follow the move of the FSP in the usual way, whereas if $p \in P_I$, the strategy Str determines the "good" move of the FSP. The only difficult point is the push operation: from $\mathsf{Push}(B, m, p', \gamma'\eta)$ Player I has to guess a tuple $C \in (\mathcal{P}(P))^{max}$ of sets of possible control states after the next pop. This is well defined if function Str is well defined, although this is a second reason why this part is not effective (even if Str is effective). ■

Corollary 2.3 *If there is a winning strategy for Player* I *in the parity game over the PDS, then there is effectively a winning pushdown strategy.*

The results in [6, 8] are in some sense stronger. One can deduce from them the winner, and a winning strategy defined by a finite automaton with output. It reads the current configuration and outputs the "next move". This strategy is positional and can also be executed by a pushdown automaton.

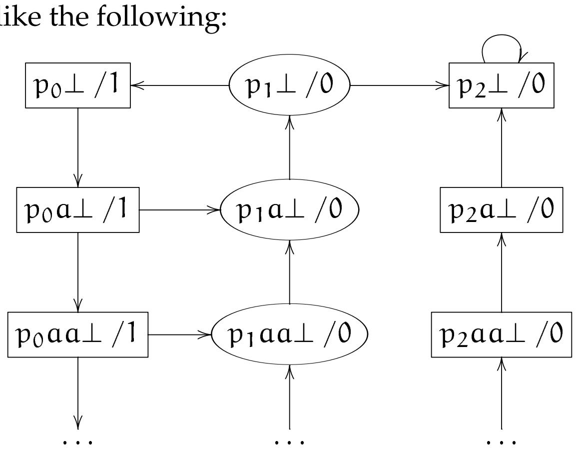
Note that in the above construction, the FSP has the same number max of priorities than the PDS, and the number of vertices is exponential in $|P|$ (more precisely, in $\mathcal{O}(|\Delta|e^{2max|P|})$). So far the best known algorithms to solve finite state parity game are polynomial in the number of vertices and exponential in the number of priorities. Applied here, we get a solution for parity games over pushdown systems (P, Γ, Δ) which is exponential in $max|P|$.

3 Example

We present here a simple example of pushdown game to illustrate the previous section. Let

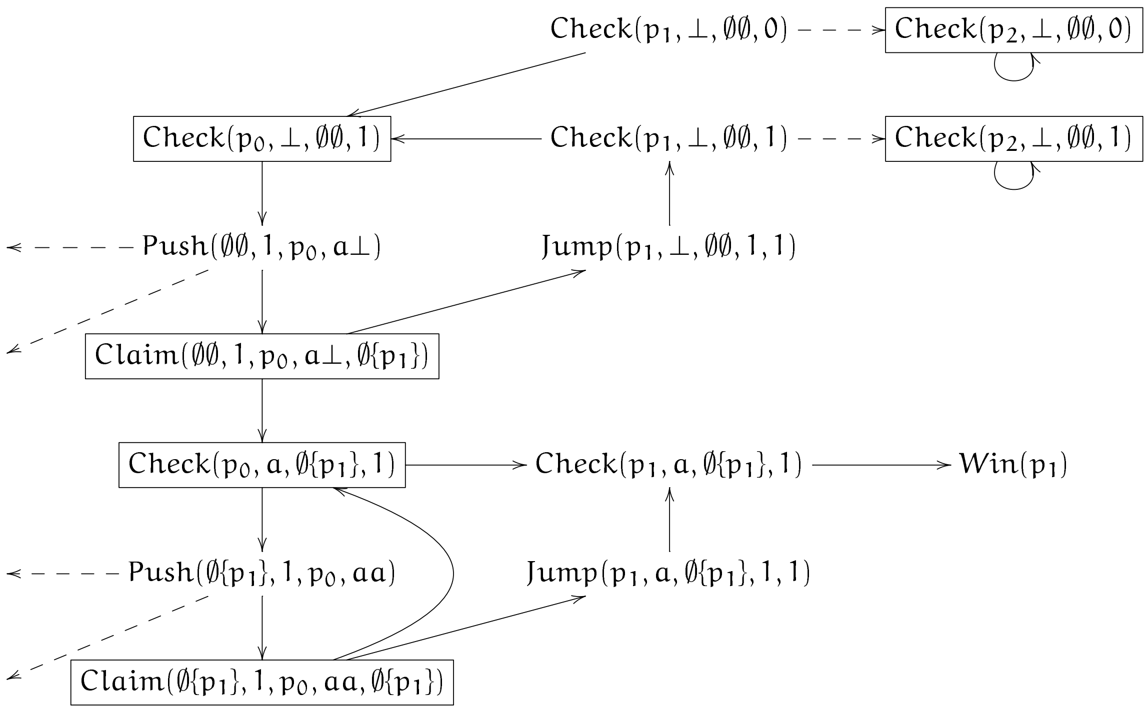
$$\Gamma = \{a, \perp\},\ P_{I} = \{p_1\},\ P_{II} = \{p_0, p_2\},$$
$$\Delta = \{(p_0, \perp, p_0, a\perp), (p_0, a, p_0, aa), (p_0, a, p_1, a), (p_1, a, p_1, \epsilon),$$
$$(p_1, \perp, p_0, \perp), (p_1, \perp, p_2, \perp), (p_2, \perp, p_2, \perp), (p_2, a, p_2, \epsilon)\},$$
$$\Omega(p_0) = 1,\ \Omega(p_1) = \Omega(p_2) = 0,\ max = 2.$$

The game graph looks like the following:



We consider the initial configuration $p_1\perp$. We represent below the part of the corresponding FSP that is relevant for Player I . Namely the solid-arrows define a winning strategy for

Player I , other arrows are dashed. We will write B_0B_1 for a tuple (B_0, B_1) in $(\mathcal{P}(P))^2$. The symbol $\perp$ cannot be removed from the stack, so the initial value of the tuple $B \in (\mathcal{P}(P))^2$ is not relevant. We set it to $\emptyset\emptyset$ in the initial vertex $\mathrm{Check}(p_1, \perp, \emptyset\emptyset, 0)$.



We see that Player I has a winning strategy from $\mathrm{Check}(p_1, \perp, \emptyset\emptyset, 0)$, by choosing always $(\emptyset, \{p_1\})$ after a Push node, and, of course, going from $p_1\perp$ to $p_0\perp$.

4 Extension to a Uniform Solution

The deficit of the result of [10] is that the winner is determined only from the initial position $p_0 \perp$. We give here an algorithm which determines the winner from any position. One need a pre-computation to solve the FSP below, e.g. with the algorithm of [9]. Moreover we get a global or “symbolic” representation of the whole winning region, which will be proved to be regular (configurations are words over the alphabet $P \cup \Gamma$).

We have seen that a configuration $p\gamma v$ of the PDS is represented in the FSP by $\mathrm{Check}(p, \gamma, B, m)$ where B “summarizes” information about v. This can be used more systematically if we know from which configurations qv Player I can win. For all $B \subseteq P$, we write $[B]^{max} = (B, \cdots, B) \in (\mathcal{P}(P))^{max}$.

Algorithm 4.1 (uniform solution for parity game on PDS)
Input: *a PDS* (P, Γ, Δ), $P = P_{\mathrm{I}} \uplus P_{\mathrm{II}}$, *and a priority function* $\Omega : P \longrightarrow \{0, \cdots, max - 1\}$, *a configuration* $\pi_0 = p\gamma_0\gamma_1 \cdots \gamma_n \in P\Gamma^*$

Output: *a winning strategy from π_0, or the answer "π_0 is not in the winning region"*

Solve first the FSP corresponding to the PDS (see proof of Theorem 2.2). Determine the winning region W_I: the set of vertices from which Player I *has a winning strategy in the FSP, and compute a positional (and uniform) winning strategy on W_I.*

$D_{n+1} := \emptyset$

for $i := n$ *downto 0* ***do***

$D_i := \{q \in P \mid \mathtt{Check}(q, \gamma_i, [D_{i+1}]^{max}, \Omega(q)) \in W_I\}$

end for

if $p \notin D_0$ ***then***

answer "π_0 is not in the winning region"

else

answer "there is a winning pushdown strategy with initial configuration $p\ \gamma_0[D_1]^{max}\Omega(p)\ \gamma_1[D_2]^{max}0\ \cdots\ \gamma_n[D_{n+1}]^{max}0$*, and transitions like in the proof of Theorem 2.2."*

end if

In the initialization of D_{n+1}, $\emptyset$ should be replaced by some $D_{n+1} \subseteq P$ if there is another convention about empty stack. More formally, this iterative computation can be transformed into an alternating automaton reading the word $p\gamma_0\gamma_1 \cdots \gamma_n$, where the transitions are defined depending on which configurations are winning in the FSP. This proves that the winning region of the PDS is regular.

Theorem 4.2 *Given a PDS with a parity winning condition, one can compute uniformly the winning region of Player* I *, which is regular, and a winning pushdown strategy with Algorithm 4.1.*

For the proof we observe that the winning condition concerns only the priorities seen infinitely often, and the result of a play does not depend on a finite prefix of it. The initialization of the strategy, as well as determining the winner, is in linear time in the length of the configuration, and the computation of the "next step" is in constant time.

It remains open how to extend the techniques of [6, 8] also to a uniform solution.

Example

We consider the same example as in section 3. If we solve completely the FSP, we see that the following nodes are in the winning region of Player I :

$$\begin{array}{lcl} \mathsf{Check}(p_0, \bot, B_0B_1, 1) & \text{for all} & (B_0, B_1) \in (\mathcal{P}(P))^2, \\ \mathsf{Check}(p_1, \bot, B_0B_1, 0) & \text{for all} & (B_0, B_1) \in (\mathcal{P}(P))^2, \\ \mathsf{Check}(p_0, a, B_0B_1, 1) & \text{if} & p_1 \in B_1, \\ \mathsf{Check}(p_1, a, B_0B_1, 0) & \text{if} & p_1 \in B_0, \\ \mathsf{Check}(p_2, a, B_0B_1, 0) & \text{if} & p_2 \in B_0. \end{array}$$

Applaying the algorithm to a configuration $\pi_0 = pa \cdots a\bot$, we get $D_{n+1} := \emptyset$ and for all $i \in \{0, n\}$ $D_i = \{p_0, p_1\}$. Then the winning region of Player I in the PDS is $\{p_0, p_1\}a^*\bot$.

5 Parity Games on Prefix-Recognizable Graphs

Among several equivalent definitions of Prefix-Recognizable Graph (class $\mathsf{REC}_{\mathsf{RAT}}$ in [2]) we choose the following. Given a finite alphabet Γ, a graph, or set of edges $G \subseteq \Gamma^* \times \Gamma^*$ is a PRG iff

$$G = \{uw \hookrightarrow vw \mid u \in U_i, v \in V_i, w \in W_i, 1 \leqslant i \leqslant N\} ,$$

where for all i, $1 \leqslant i \leqslant N$, the U_i, V_i, W_i are regular sets over Γ.

Games over PRG are defined in a natural way. In a configuration $x \in \Gamma^+$, the first letter determines the priority and the player whose turn it is. For technical reasons the priority function is $\Omega : \Gamma \longrightarrow \{2, \cdots, max - 1\}$, extended to Γ^+ by $\Omega(ax) = \Omega(a), \forall a \in \Gamma, x \in \Gamma^*$. And we have $\Gamma = \Gamma_{\mathrm{I}} \uplus \Gamma_{\mathrm{II}}$, $V_{\mathrm{I}} = \Gamma_{\mathrm{I}}\Gamma^*$, and $V_{\mathrm{II}} = \Gamma_{\mathrm{II}}\Gamma^*$, similarly to the PDS. A game starting from $\pi_0 \in \Gamma^+$ is defined in the usual way. Again we consider max-parity: I wins $\pi_0\pi_1 \cdots$ iff $\limsup_{k\to\infty} \Omega(\pi_k)$ is odd.

Reduction to Parity Game on Pushdown Graph

We will define a PDS (P, Γ', Δ) which is equivalent to the PRG in the sense that Player I wins the PDS iff he wins the PRG, and a winning strategy in one game can be effectively constructed from a winning strategy in the other game.

Let $\Gamma' = \Gamma \uplus \{\bot\}$. A vertex $ax \in \Gamma^+$ of the PRG ($a \in \Gamma$) is represented by the configuration $t_k^{\mathrm{I}}ax\bot$ or $t_k^{\mathrm{II}}ax\bot$, if $k = \Omega(a)$ and a is in Γ_{I} or Γ_{II} respectively ($t_k^{\mathrm{I}}, t_k^{\mathrm{II}} \in P$). The idea of the reduction is to decompose the transition $uw \hookrightarrow vw$ of the PRG letter by letter, using intermediate configurations in the PDS.

Theorem 5.1 *Given a PRG with parity condition, one can construct in linear time a PDS which is equivalent to the PRG in the following sense: a play over the PRG is mapped to a play over the PDS*

preserving the winning condition. Consequently:
1. the winner of the parity-PRG from a given configuration is the winner of the parity-PDS from the corresponding configuration,
2. a winning strategy in the parity-PRG can be calculated from a winning strategy in the parity-PDS.

Proof: Each regular set is recognized by a (say deterministic, complete) finite automaton: $\mathcal{B}_i$ for U_i, $\mathcal{C}_i$ for $\tilde{V}_i$, $\mathcal{D}_i$ for W_i. Here $\tilde{V}_i$ is the mirror language of V_i, *i.e.*, $\mathcal{C}_i$ is reading from right to left. We note p_{ij} the states of $\mathcal{B}_i$, q_{ij} and r_{ij} those of $\mathcal{C}_i$ and $\mathcal{D}_i$ respectively. The corresponding initial states are p_{i0}, q_{i0}, and r_{i0}. We note $p_{ij} \xrightarrow{a} p_{ij'}$ a transition in $\mathcal{B}_i$ labeled by the letter a. In the following j is always an integer in the *finite* range $[0, NS]$ where NS is the maximal number of states of the automata $\mathcal{B}_i$, $\mathcal{C}_i$, and $\mathcal{D}_i$.

We define now the PDS that simulates the PRG. The control-states of the PDS have the same names as the states of the automata $\mathcal{B}_i$, $\mathcal{C}_i$, and $\mathcal{D}_i$, with additional superscript I or II (i is ranging over $[1, N]$):

$$\begin{aligned} P_I &= \{p^I_{ij} \mid 0 \leqslant j \leqslant NS\} \cup \{t^I_k \mid 2 \leqslant k < max\} \cup \{s^I_i\} \\ P_{II} &= \{p^{II}_{ij} \mid 0 \leqslant j \leqslant NS\} \cup \{t^{II}_k \mid 2 \leqslant k < max\} \cup \{s^{II}_i\}\,. \end{aligned}$$

Additional control states t^I_k and t^{II}_k are used to mark the configurations of the PDS that correspond to vertices of the PRG. States s^I_i, s^{II}_i are added for technical reasons.

The transitions rules of the PDS are the following: for all $a, b \in \Gamma, c \in \Gamma'$,

$t^I_k c \hookrightarrow p^I_{i0} c$ (Player I chooses to use in the PRG a transition of type i: $u_i w_i \hookrightarrow v_i w_i$ $u_i \in U_i, v_i \in V_i, w_i \in W_i$),

$t^{II}_k c \hookrightarrow p^{II}_{i0} c$ (similarly for Player II).

Then for all $\sigma \in \{I, II\}$,

$p^\sigma_{ij} a \hookrightarrow p^\sigma_{ij'}$, if $p_{ij} \xrightarrow[\mathcal{B}_i]{a} p_{ij'}$ ("reading" of u_i),

$p^\sigma_{ij} c \hookrightarrow s^{\bar{\sigma}}_i c$ if p_{ij} is a final state of $\mathcal{B}_i$ (Player σ decides that the word u_i ends here, and asks the opponent for agreement).

The opponent of σ is denoted $\bar{\sigma}$.

$s^{\bar{\sigma}}_i c \hookrightarrow r^\sigma_{i0} c$ (the opponent wants to verify that the rest of the stack is really in W_i, because he thinks that this is not the case)

$r^\sigma_{ij} a \hookrightarrow r^\sigma_{ij'}$, if $r_{ij} \xrightarrow[\mathcal{D}_i]{a} r_{ij'}$ ("reading" of w_i, then:)

$r^\sigma_{ij}\perp$ is immediate lost for σ if r^σ_{ij} is not final in $\mathcal{D}_i$,
and immediate win for σ if r^σ_{ij} is final in $\mathcal{D}_i$,

$s^{\bar{\sigma}}_i c \hookrightarrow q^\sigma_{i0} c$ (otherwise, the opponent is trusting Player σ, and lets him continue),

$q^\sigma_{ij} c \hookrightarrow q^\sigma_{ij'} bc$ if $q_{ij} \xrightarrow[\mathcal{C}_i]{b} q_{ij'}$ ("writing" of v_i, chosen by Player σ),

$q^{\sigma}_{ij}a \hookrightarrow t^{I}_{k}a$ if q_{ij} is a final state of $\mathcal{C}_i$, $a \in \Gamma_I$ and $k = \Omega(a)$

(Player σ chooses that ν_i ends here),

$q^{\sigma}_{ij}a \hookrightarrow t^{II}_{k}a$ if q_{ij} is a final state of $\mathcal{C}_i$, $a \in \Gamma_{II}$ and $k = \Omega(a)$ (similarly).

Note that the control state t^{I}_{k} is redundant with the first letter.

Of course the priority of t^{σ}_{k} is $\Omega(t^{\sigma}_{k}) = k$. Given $x \in \Gamma^+$, it is clear that: $x \hookrightarrow y$ in the PRG ($y \in \Gamma^*$) iff from the corresponding configuration $t^{\sigma}_{k}x\perp$, Player σ can reach the configuration $t^{\sigma'}_{k}y\perp$ corresponding to y or wins immediately (if the opponent $\bar{\sigma}$ thinks that σ wants to violate the transition rule).

Now the unique deficit of this construction is that Player σ can stay forever in the states q^{σ}_{ij}, pushing infinitely many new letters onto the stack. To avoid this unfair behavior, which does not correspond to a real transition of the PRG, we define the following priorities: $\Omega(p^{I}_{ij}) = \Omega(q^{I}_{ij}) = \Omega(r^{I}_{ij}) = \Omega(s^{I}_{i}) = 0$, $\Omega(p^{II}_{ij}) = \Omega(q^{II}_{ij}) = \Omega(r^{II}_{ij}) = \Omega(s^{II}_{i}) = 1$. These priorities are lower than the normal priorities, so they have no influence on the winning condition of the "real" game. One takes 0 for Player I , while Player I wants an odd number; so he can't win by staying in those intermediate states. Similarly for Player II . ■

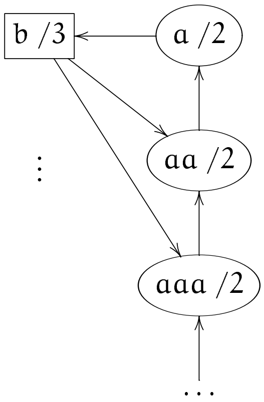
The reduction presented here is not so far from the result of [3] (Proposition 4.2 of the full version), that the prefix recognizable graphs are obtained from the pushdown graphs by ϵ-closure. It should be possible to extend the symbolic solution of [1] for reachability and Büchi games on pushdown graphs to prefix-recognizable graphs (with the new feature of optimal strategy).

Example

Let $\Gamma = \{a, b\}$, we consider the following PRG:

$$G = (a \hookrightarrow \epsilon)a^+ \cup (a \hookrightarrow b)\epsilon \cup (b \hookrightarrow a^+)\epsilon ,$$

writen here in the form $(U_1 \hookrightarrow V_1)W_1 \cup (U_2 \hookrightarrow V_2)W_2 \cup (U_3 \hookrightarrow V_3)W_3$, with $N = 3$. Let $max = 4$, $\Omega(a) = 2$, $\Omega(b) = 3$, $\Gamma_I = \{a\}$, $\Gamma_{II} = \{b\}$. The game graph is pictured here:



According to the above construction, a vertex a^i is represented in the PDS by $t^{\mathrm{I}}_2 a^i \bot$, and b is represented by $t^{\mathrm{II}}_3 b\bot$. The automaton recognizing U_i, V_i and W_i are very simple, we do not define them explicitely. We draw a part of the graph of the corresponding PDS:

$$
\begin{tikzcd}
p^{\mathrm{II}}_{31}\bot \arrow[d] & p^{\mathrm{II}}_{30}b\bot \arrow[l] & \boxed{t^{\mathrm{II}}_3 b\bot} \arrow[l] & q^{\mathrm{I}}_{21}b\bot \arrow[l] & q^{\mathrm{I}}_{20}\bot \arrow[l] & s^{\mathrm{II}}_2\bot \arrow[l] & \\
s^{\mathrm{I}}_3\bot \arrow[r] \arrow[d, dashed] & q^{\mathrm{II}}_{30}\bot \arrow[r] & q^{\mathrm{II}}_{31}a\bot \arrow[r] \arrow[ddd] & t^{\mathrm{I}}_2 a\bot \arrow[r] & p^{\mathrm{I}}_{20}a\bot \arrow[r] & p^{\mathrm{I}}_{21}\bot \arrow[u] & \\
{} & & & & q^{\mathrm{I}}_{11}a\bot \arrow[ul] & s^{\mathrm{II}}_1 a\bot \arrow[l] \arrow[r, dashed] & {} \\
& & & & & & \\
& & q^{\mathrm{II}}_{31}aa\bot \arrow[r] \arrow[dd] & t^{\mathrm{I}}_2 aa\bot \arrow[r] & p^{\mathrm{I}}_{10}aa\bot \arrow[r] & p^{\mathrm{I}}_{11}a\bot \arrow[uu] & \\
& & & & q^{\mathrm{I}}_{11}aa\bot \arrow[ul] & s^{\mathrm{II}}_1 aa\bot \arrow[l] \arrow[r, dashed] & {} \\
& & q^{\mathrm{II}}_{31}aaa\bot \arrow[r] \arrow[d] & t^{\mathrm{I}}_2 aaa\bot \arrow[r] & p^{\mathrm{I}}_{10}aaa\bot \arrow[r] & p^{\mathrm{I}}_{11}aa\bot \arrow[u] & \\
& & \cdots & & \cdots \arrow[ul] & &
\end{tikzcd}
$$

Player I has a winning strategy from b in the PRG.

Discussion

It remains open how to apply the MSO-definability of a winning region (either for deciding the winner or for extracting a winning strategy). Another question is to develop a theory of game simulation which covers the examples of Sections 2 and 5.

Acknowledgment

Great thanks the referees of ICALP [1] for suggesting the idea of Section 4, to Igor Walukiewicz for helping me understand his paper, and to Wolfgang Thomas, Christof Löding and the referees of INFINITY for helpful remarks.

References

[1] T. CACHAT, *Symbolic Strategy Synthesis for Games on Pushdown Graphs*, ICALP'02, LNCS 2380, pp. 704-715, 2002. available at `http://www-i7.informatik.rwth-aachen.de/~cachat/`.

[2] D. CAUCAL, *On infinite transition graphs having a decidable monadic theory*, ICALP'96, LNCS 1099, pp. 194–205, 1996.

[3] D. CAUCAL, *On the transition graphs of Turing machines*, 3rd MCU, LNCS 2055, pp. 177-189, 2001. Full version to appear in TCS, available at `http://www.irisa.fr/prive/caucal/biblio.html`.

[4] E. A. EMERSON and C. S. JUTLA, *Tree automata, mu-calculus and determinacy*, FoCS'91, IEEE Computer Society Press, pp. 368–377, 1991.

[5] E. A. EMERSON, C. S. JUTLA, and A. P. SISTLA, *On model-checking for fragments of* μ-*calculus*, CAV'93, LNCS 697, pp. 385–396, 1993.

[6] O. KUPFERMAN and M. Y. VARDI, *An Automata-Theoretic Approach to Reasoning about Infinite-State Systems*, CAV'00, LNCS 1855, pp. 36–52, 2000.

[7] O. SERRE, *Note on Winning Positions on Pushdown Games with Omega-Regular Conditions*, submited to Information Processing Letter, 2002.

[8] M. Y. VARDI, *Reasoning about the past with two-way automata*, ICALP'98, LNCS 1443, pp. 628–641, 1998.

[9] J. VÖGE AND M. JURDZIŃSKI, *A discrete strategy improvement algorithm for solving parity games*, CAV'00, LNCS 1855 pp. 202–215, 2000.

[10] I. WALUKIEWICZ, *Pushdown processes: games and model checking*, CAV'96, LNCS 1102, pp. 62–74, 1996. Full version in Information and Computation 157, 2000.