Playing Pushdown Parity Games in a Hurry

Wladimir Fridman
Chair of Computer Science 7
RWTH Aachen University
Aachen, Germany
fridman@automata.rwth-aachen.de

Martin Zimmermann
Institute of Informatics
University of Warsaw
Warsaw, Poland
zimmermann@mimuw.edu.pl

We continue the investigation of finite-duration variants of infinite-duration games by extending known results for games played on finite graphs to those played on infinite ones. In particular, we establish an equivalence between pushdown parity games and a finite-duration variant. This allows us to determine the winner of a pushdown parity game by solving a reachability game on a finite tree.

1 Introduction

Infinite two-player games on graphs are a powerful tool to model, verify, and synthesize open reactive systems and are closely related to fixed-point logics. The winner of a play in such a game typically emerges only after completing the whole (infinite) play. Despite this, McNaughton became interested in playing infinite games in finite time, motivated by his belief that "infinite games might have an interest for casual living room recreation" [6].

As playing infinitely long is impossible for human players, McNaughton introduced scoring functions for Muller games, a certain type of infinite game. Each of these functions is associated to one of the two players, so it makes sense to talk about the scores of a player. The scoring functions are updated after every move and describe the progress a player has made towards winning the play. However, as soon as a scoring function reaches its predefined threshold, the game is stopped and the player whose score reached its threshold first is declared to win this (now finite) play.

On the theoretical side, by applying finite-state determinacy of Muller games, McNaughton showed that a Muller game and a finite-duration variant with a factorial threshold score have the same winner. Thus, the winner of a Muller game can be determined by solving a finite reachability game, which is much simpler to solve, albeit doubly-exponentially larger than the original Muller game.

This result was improved by showing that the finite-duration game with threshold three always has the same winner as the original Muller game [3] and by a (score-based) reduction from a Muller game to a safety game whose solution not only yields the winner of the Muller game, but also a winning strategy [8]. The improved threshold does not rely on finite-state determinacy, but is proven by constructing strategies that are winning for both games at the same time.

The reduction from Muller to safety games yields a new memory structure for Muller games that implements not only a winning strategy, but also the most general non-deterministic winning strategy (a so-called permissive strategy) that prevents the losing player from reaching a certain score. This extends the work of Bernet et al. on permissive strategies for parity games [1] to Muller games. For parity games, the algorithm presented to compute a permissive strategy is Jurdziński's progress measure algorithm [4] for solving parity games. This raises the question of whether there is also a (score-based) progress measure algorithm for Muller games, which can be derived from the construction of a permissive strategy.

In this work, we begin to extend these results to infinite games played on infinite game graphs. At first, two questions have to be answered: what type of infinite game graphs and what type of winning

M. Faella, A. Murano (Eds.): Games, Automata, Logics
and Formal Verification (GandALF 2012)
EPTCS 96, 2012, pp. 183–196, doi:10.4204/EPTCS.96.14

condition to consider? We have to restrict the type of game graphs, since stopping a play after a finite number of rounds can only lead to an equivalent finite-duration variant if there is some regularity in the game graph. A well-researched class of infinite graphs are configuration graphs of pushdown systems. Walukiewicz showed how to solve parity games on such game graphs in exponential time by a reduction to parity games on finite game graphs [9].

As for the second question, we also consider parity games. In his work on making infinite games playable for human players, McNaughton was interested in Muller games, since he thought that games for human players should not be positionally determined in order to be interesting enough. From a theoretical point of view, this can be argued as follows: every positionally determined game in a finite game graph has a trivial finite-duration variant. In this variant, a play is stopped as soon as a vertex is visited for the second time and the winner is the player who wins the infinite play induced by this cycle. As every positional winning strategy for the infinite-duration game is also winning for the finite-duration game, the two games have the same winner.

For a (min-) parity game on a finite game graph this criterion can be improved: let $|V|_c$ denote the number of vertices colored by c. Then, a positional winning strategy for Player $i \in \{0,1\}$ does not visit $|V|_c + 1$ vertices of color c with parity $1-i$ without visiting a vertex of smaller color in between. This condition can be expressed using scoring functions Sc_c that count the number of vertices of color c visited since the last visit of a vertex of color $c' < c$. Due to positional determinacy of parity games, the following finite-duration game has the same winner as the original parity game: a play is stopped as soon as some scoring function Sc_c reaches value $|V|_c + 1$ for the first time and Player i is declared to be the winner, if the parity of c is i. Again, a positional winning strategy for the parity game is also winning for the finite-duration game, i.e., the two games indeed have the same winner.

However, both criteria do not necessarily yield a finite-duration game when applied to a game on an infinite game graph; the first one since there could be infinite simple paths, the second one since there are colors that color infinitely many vertices, i.e., $|V|_c$ could be infinite. Hence, devising a finite-duration variant of games on infinite game graphs requires more sophisticated criteria, even if the game is positionally determined.

We exploit the intrinsic structure of the game graph induced by the pushdown system by defining stair-score functions $\mathsf{StairSc}_c$ for every color c and show the equivalence between a parity game and the finite-duration version, when played up to an exponential threshold stair-score (in the size of the pushdown system). This result shows how to determine the winner of an infinite game on an infinite game graph by solving a finite reachability game. We complement this by giving a lower bound on the threshold stair-score that always yields the same winner, which is exponential in the cubic root of the size of the underlying pushdown system.

To prove our main theorem, we analyze Walukiewicz's reduction from parity games on pushdown graphs to parity games on finite graphs and prove a correspondence between stair-scores in the pushdown game and scores in the finite parity game. The winning player of the finite parity game (who also wins the pushdown game) has a winning strategy that bounds the losing player's scores by $|V|_c$ (the number of vertices colored by c in the finite parity game). We show that this strategy can be turned into a winning strategy for him in the pushdown game that bounds the stair-scores by $|V|_c$ as well. Since the finite parity game is of exponential size, our result follows.

This work is organized as follows: after fixing our notation for parity games and pushdown systems in Section 2, we introduce the score and stair-score functions in Section 3. In Section 4, we recall Walukiewicz's reduction, which we apply in Section 5 to prove our main theorem, namely the equivalence between parity games on pushdown graphs and their finite-duration variant. Finally, in Section 6, we prove the lower bounds on the threshold score that always yields an equivalent finite-duration game.

2 Preliminaries

The power set of a set X is denoted by $\mathscr{P}(X)$. The set of non-negative integers is denoted by $\mathbb{N}$. For $n \in \mathbb{N}$, let $[n] = \{0, \dots, n-1\}$ and $\mathrm{Par}(n) = 0$ if n is even, and $\mathrm{Par}(n) = 1$ if n is odd. Moreover, for every alphabet Σ, i.e., a finite set of symbols, the set of finite words is denoted by Σ^*, and Σ^ω denotes the set of infinite words. The length of a word $w \in \Sigma^*$ is denoted by $|w|$ and ε denotes the empty word, i.e., the word of length $|\varepsilon| = 0$. For $n \in \mathbb{N}$, the set of words of length at most n is denoted by $\Sigma^{\leq n}$ and for $\Sigma^* \setminus \{\varepsilon\}$ we also write Σ^+. For a word $w \in \Sigma^+ \cup \Sigma^\omega$ and $n \in \mathbb{N}$, we write $w(n-1)$ for the n-th letter of w (the first letter is $w(0)$) and denote its last letter by $\mathsf{last}(w)$. For $w \in \Sigma^*$ and $w' \in \Sigma^* \cup \Sigma^\omega$, we write $w \sqsubseteq w'$ if w is a prefix of w' and $w \sqsubset w'$ if w is a strict prefix of w'. For a word $\rho \in \Sigma^\omega$, let $\mathrm{Inf}(\rho) = \{a \in \Sigma \mid \rho(n) = a \text{ for infinitely many } n\}$.

2.1 Parity Games

A game graph is a tuple $G = (V, V_0, V_1, E, v_{\mathrm{in}})$ where (V, E) is a (possibly countably infinite) directed graph with set V of vertices and set $E \subseteq V \times V$ of edges, where $V_0 \cup V_1$ is a partition of V and $v_{\mathrm{in}} \in V$ is the initial vertex. We assume that every vertex has at least one outgoing edge. Vertices from V_i belong to Player i, for $i \in \{0, 1\}$.

A parity game $\mathscr{G} = (G, \mathrm{col})$ consists of a game graph G and a coloring function $\mathrm{col}\colon V \to [n]$, for some $n \in \mathbb{N}$. Given col, we define $\mathsf{MinCol}\colon V^+ \to [n]$ by $\mathsf{MinCol}(w) = \min\{\mathrm{col}(w(i)) \mid 0 \leq i < |w|\}$. A play of $\mathscr{G}$ is built up by the two players by moving a token on the game graph. Initially, the token is placed on v_{in}. In every round, if the current vertex v is in V_i, then Player i has to choose an outgoing edge $(v, v') \in E$ and the token is moved to the successor v'. Thus, a play in $\mathscr{G}$ is an infinite sequence $\rho \in V^\omega$ such that $\rho(0) = v_{\mathrm{in}}$ and $(\rho(n), \rho(n+1)) \in E$ for every $n \in \mathbb{N}$. Such a play ρ is winning for Player 0 if $\min\{\mathrm{Inf}(\mathrm{col}(\rho))\}$ is even, otherwise it is winning for Player 1. Here, $\mathrm{col}(\rho)$ represents the sequence of colors seen by ρ. Thus, we sometimes refer to the coloring function col as a min-parity condition.

A strategy for Player i is a function $\sigma\colon V^*V_i \to V$ such that $(\mathsf{last}(w), \sigma(w)) \in E$ for every $w \in V^*V_i$. A strategy σ is called positional if $\sigma(w) = \sigma(w')$ holds for all $w, w' \in V^*V_i$ with $\mathsf{last}(w) = \mathsf{last}(w')$. A play ρ is consistent with σ for Player i if $\rho(n+1) = \sigma(\rho(0) \cdots \rho(n))$ for every $n \in \mathbb{N}$ with $\rho(n) \in V_i$. A strategy σ is a winning strategy for Player i if every play ρ that is consistent with σ is winning for Player i. We say that Player i wins $\mathscr{G}$ if there exists a winning strategy for Player i. A game is determined if one of the players wins it.

Theorem 1 ([2, 7])**.** *Parity games are determined with positional winning strategies.*

2.2 Pushdown Game Graphs

A pushdown system (PDS) $\mathscr{P} = (Q, \Gamma, \Delta, q_{\mathrm{in}})$ consists of a finite set of states Q with an initial state $q_{\mathrm{in}} \in Q$, a stack alphabet Γ with the initial stack symbol $\bot \notin \Gamma$, which can neither be written nor deleted from the stack, and a transition relation $\Delta \subseteq Q \times \Gamma_\bot \times Q \times \Gamma_\bot^{\leq 2}$, where $\Gamma_\bot = \Gamma \cup \{\bot\}$. We say that a transition $\delta = (q, A, q', \alpha) \in \Delta$ is a push-transition if $|\alpha| = 2$, δ is a skip-transition if $|\alpha| = 1$, and δ is a pop-transition if $\alpha = \varepsilon$. In the following, we assume every PDS to be deadlock-free, i.e., for every $q \in Q$ and $A \in \Gamma_\bot$ there exist $q' \in Q$ and $\alpha \in \Gamma_\bot^{\leq 2}$ such that $(q, A, q', \alpha) \in \Delta$.

A stack content is a word from $\Gamma^*\bot$ where the leftmost symbol is assumed to be the top of the stack. A configuration is a pair (q, γ) consisting of a state $q \in Q$ and a stack content $\gamma \in \Gamma^*\bot$. The stack height of a configuration (q, γ) is defined by $\mathrm{sh}(q, \gamma) = |\gamma| - 1$. Furthermore, we write $(q, \gamma) \vdash (q', \gamma')$ if there exists $(q, \gamma(0), q', \alpha) \in \Delta$ and $\gamma' = \alpha\gamma(1) \cdots \gamma(|\gamma| - 1)$.

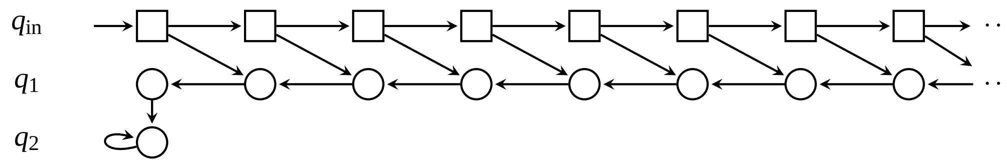


Figure 1: A pushdown game graph (only the part reachable from the initial vertex is shown)

For a PDS $\mathscr{P}$, the induced pushdown graph is the infinite directed graph $G(\mathscr{P}) = (V,E)$ where $V = \{(q,\gamma) \mid q \in Q, \gamma \in \Gamma^*\bot\}$ is the set of configurations and $(v,v') \in E$ if $v \vdash v'$. Notice that every vertex of the pushdown graph $G(\mathscr{P})$ has at least one outgoing edge, since $\mathscr{P}$ is deadlock-free. Consider a partition $Q_0 \cup Q_1$ of the set of states Q. The induced pushdown game graph $G = (V, V_0, V_1, E, v_{\text{in}})$ is a game graph where $(V,E) = G(\mathscr{P})$, the partition $V_0 \cup V_1$ of the set of configurations V is defined by $V_i = \{(q,\gamma) \in V \mid q \in Q_i\}$, for $i \in \{0,1\}$, and $v_{\text{in}} = (q_{\text{in}}, \bot)$. Given such a pushdown game graph G and a coloring $\mathrm{col}\colon Q \to [n]$ of its states, we obtain a parity game by extending col to configurations via $\mathrm{col}(q,\gamma) = \mathrm{col}(q)$, for every state $q \in Q$ and every stack content $\gamma \in \Gamma^*\bot$. We refer to such a game as a pushdown game.

Example 1. *Consider the pushdown system* $\mathscr{P} = (\{q_{\text{in}}, q_1, q_2\}, \{A\}, \Delta, q_{\text{in}})$ *where* Δ *is the following set*

$$\{(q_{\text{in}}, X, q_{\text{in}}, AX), (q_{\text{in}}, X, q_1, AX), (q_1, A, q_1, \varepsilon), (q_1, \bot, q_2, \bot), (q_2, A, q_2, \varepsilon), (q_2, \bot, q_2, \bot) \mid X \in \{A, \bot\}\}.$$

The partition $Q_0 = \{q_1, q_2\}$ *and* $Q_1 = \{q_{\text{in}}\}$ *yields the pushdown game graph G depicted in Figure 1, where the circles indicate Player* 0 *configurations and squares are Player* 1 *configurations. With the coloring function* col *such that* $\mathrm{col}(q_{\text{in}}) = \mathrm{col}(q_2) = 0$ *and* $\mathrm{col}(q_1) = 1$ *Player* 0 *wins the pushdown game* (G, col), *as every play visits only a finite number of configurations colored by* 1.

We extend the notion of PDS to pushdown transducers (PDT) by attaching input and output alphabets. A PDT $\mathscr{T} = (Q, \Gamma, \Delta, q_{\text{in}}, \Sigma_I, \Sigma_O, \lambda)$, where Q, Γ and q_{in} are as for PDS and Δ is modified such that $\Delta \subseteq Q \times \Gamma_\bot \times (\Sigma_I \cup \{\varepsilon\}) \times Q \times \Gamma_\bot^{\leq 2}$, additionally contains an input alphabet Σ_I, an output alphabet Σ_O, and a partial output function $\lambda\colon Q \to \Sigma_O$. A PDT is deterministic if it satisfies

$$\left|\{(q',\alpha) \mid (q,A,a,q',\alpha) \in \Delta\}\right| + \left|\{(q',\alpha) \mid (q,A,\varepsilon,q',\alpha) \in \Delta\}\right| \leq 1$$

for all $q \in Q$, all $a \in \Sigma_I$, and all $A \in \Gamma_\bot$. Analogously to PDS, we write $(q,\gamma) \overset{a}{\vdash} (q',\gamma')$ if there exists a transition $(q, \gamma(0), a, q', \alpha) \in \Delta$ such that $\gamma' = \alpha\gamma(1)\cdots\gamma(|\gamma|-1)$. A run ρ of a PDT on a word $w \in (\Sigma_I)^*$ is a sequence of configurations $\rho = (q_0,\gamma_0)\cdots(q_m,\gamma_m)$ such that $\rho(0) = (q_{\text{in}}, \bot)$, for all $0 \leq i < m$ there exists $a_i \in \Sigma_I \cup \{\varepsilon\}$ with $(q_i,\gamma_i) \overset{a_i}{\vdash} (q_{i+1},\gamma_{i+1})$ such that $a_0 \cdots a_{m-1} = w$, and $\{(q,\alpha) \mid (q_m, \gamma_m(0), \varepsilon, q, \alpha) \in \Delta\}$ is empty (i.e., no execution of an ε-transition is possible from the last configuration of a run). A deterministic PDT $\mathscr{T}$ defines a partial function $f_{\mathscr{T}}\colon (\Sigma_I)^* \to \Sigma_O$ such that $f_{\mathscr{T}}(w) = \lambda(q)$, where q is the state of the last configuration of the (unique) run of $\mathscr{T}$ on w, if such a run exists.

To implement pushdown strategies in a pushdown game we will use PDT. To have a finite input alphabet, we represent play prefixes here by sequences of transitions and not by sequences of configurations. Notice that both representations can easily be converted into each other. Furthermore, the output will be the next transition to be chosen by Player i instead of the next configuration. Hence, we use the set of transitions of the PDS defining the pushdown game for both the input and the output alphabet of the PDT. So, the transducer consumes a play prefix in the pushdown graph represented by a sequence of transitions and outputs the transition which Player i should choose next (in case the last configuration of the play prefix is a Player i configuration). Thus, we have to require the output transition to be executable from the last configuration of the play prefix induced by the input sequence.

3 Finite-Time Pushdown Games

In this section, we introduce a finite-duration variant of pushdown games. To this end, we adapt the concept of scoring functions, which were originally introduced by McNaughton [6] for Muller games (see also [3]), to parity games. In the following, let (G,col) be a parity game with $G = (V,V_0,V_1,E,v_{\mathrm{in}})$ and $\mathrm{col}\colon V \to [n]$.

Definition 2 (Scoring functions)**.** *For every $c \in [n]$, define the function $\mathsf{Sc}_c\colon V^* \to \mathbb{N}$ by $\mathsf{Sc}_c(\varepsilon) = 0$ and for $w \in V^*$ and $v \in V$ by*

$$\mathsf{Sc}_c(wv) = \begin{cases} \mathsf{Sc}_c(w) & \text{if } \mathrm{col}(v) > c, \\ \mathsf{Sc}_c(w)+1 & \text{if } \mathrm{col}(v) = c, \\ 0 & \text{if } \mathrm{col}(v) < c. \end{cases}$$

Furthermore, for every $c \in [n]$, $\mathsf{MaxSc}_c\colon V^* \cup V^\omega \to \mathbb{N} \cup \{\infty\}$ is defined by $\mathsf{MaxSc}_c(\rho) = \max_{w \sqsubseteq \rho} \mathsf{Sc}_c(w)$.

A positional winning strategy for Player i in a parity game does not visit a vertex v with $\mathrm{Par}(\mathrm{col}(v)) = 1-i$ twice without visiting some vertex of strictly smaller color in between. Hence, applying the pigeonhole principle shows that positional winning strategies in finite parity games bound the scores of the losing player. For $c \in [n]$, let $|V|_c$ denote the number of vertices of color c, i.e., $|V|_c = |\{v \in V \mid \mathrm{col}(v) = c\}|$.

Remark 3. *Let σ be a positional winning strategy for Player i in a parity game with a finite vertex set V. Then, for every ρ that is consistent with σ, $\mathsf{MaxSc}_c(\rho) \leq |V|_c$ for all $c \in [n]$ such that $\mathrm{Par}(c) = 1-i$.*

Thus, winning a finite parity game, i.e., a parity game with a finite game graph, can also be characterized by being able to achieve a certain threshold score. As soon as this threshold score is reached the play can be stopped, since the winner is certain. This is the idea behind finite-time versions of infinite games. Formally, a finite-time parity game (G,col,k) consists of a game graph G, a min-parity condition col and a threshold $k \in \mathbb{N} \setminus \{0\}$. A play in (G,col,k) is a finite path $w = w(0)\cdots w(r) \in V^*$ with $w(0) = v_{\mathrm{in}}$ such that $\mathsf{MaxSc}_c(w) = k$ for some $c \in [n]$, and $\mathsf{MaxSc}_c(w(0)\cdots w(r-1)) < k$ for all $c \in [n]$. The play w is winning for Player i if $\mathrm{Par}(c) = i$. The notions of (winning) strategies are defined as usual.

By induction over ther number n of colors one can show that every threshold k is eventually reached by some score function if the path is sufficiently long. Thus, there are no draws due to infinite plays.

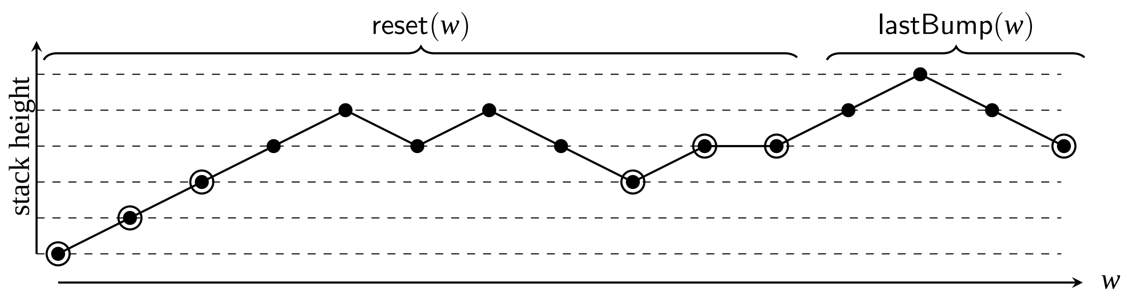
Lemma 4. *For every $w \in V^*$ with $|w| \geq k^n$, there is some $c \in [n]$ such that $\mathsf{MaxSc}_c(w) \geq k$.*

Hence, a play in a finite-time parity game is stopped after at most exponentially many rounds. Moreover, using the construction of [3] for Muller games (which also holds for parity games) it can also be shown that the bound in Lemma 4 is tight, i.e., for every k there is a $w \in V^*$ with $|w| = k^n - 1$ such that $\mathsf{MaxSc}_c(w) < k$ for all $c \in [n]$.

Furthermore, it is never the case that two different score functions are increased in the same round: by definition of the score functions, only the value of $\mathsf{Sc}_{\mathrm{col}(w(i))}$ is increased in round i of a play w. Hence, as soon as some score function is increased to the threshold a unique winner can be declared.

Lemma 5. *Let $w \in V^*$, $v \in V$ and $c, c' \in [n]$. If $\mathsf{Sc}_c(wv) = \mathsf{Sc}_c(w)+1$ and $\mathsf{Sc}_{c'}(wv) = \mathsf{Sc}_{c'}(w)+1$, then $c = c'$.*

In [3], the equivalence between Muller games and finite-time Muller games (using the original scoring functions for Muller games) on finite game graphs is shown for the constant threshold $k = 3$. A simple consequence of Remark 3 is an analogous result for parity games on finite game graphs.



col	0	2	1	0	2	1	0	0	0	1	1	1	1	2	1
$\mathsf{StairSc}_0$	1	1	1	2	2	2	3	3	2	2	2	2	2	2	2
$\mathsf{StairSc}_1$	0	0	1	0	0	1	0	0	0	1	2	3	4	4	3
$\mathsf{StairSc}_2$	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 2: A finite path w, its stair positions and its stair-scores.

Theorem 6. *Let G be a finite game graph with vertex set V and $\mathrm{col}\colon V \to [n]$. For every threshold $k > \max_{c\in[n]} |V|_c$, Player i wins (G, col) if and only if Player i wins (G, col, k).*

It is easy to see that this result does not hold for infinite game graphs. Consider the pushdown game from Example 1 and recall that Player 0 wins it. However, for every threshold $k > 0$, Player 1 has a winning strategy in the corresponding finite-time pushdown game by moving the token to configuration $(q_1, A^{k-1}\bot)$, which completely specifies a strategy for Player 1. Following this strategy, Player 1 wins since color 1 is the first to reach score k which happens when the token arrives at the configuration $(q_1, \bot)$.

To obtain an analogous result for pushdown games, we have to adapt the scoring functions. Now, let (G, col) be a pushdown game. Fix a path through the pushdown graph. A configuration is said to be a stair configuration, if no subsequent configuration of smaller stack height exists in this path.

Definition 7 (Stairs [5])**.** *Define the functions $\mathsf{StairPositions}\colon V^+ \cup V^\omega \to 2^{\mathbb{N}}$ and $\mathsf{Stairs}\colon V^+ \cup V^\omega \to V^+ \cup V^\omega$ as follows: for $w \in V^+ \cup V^\omega$, let*

$$\mathsf{StairPositions}(w) = \{n \in \mathbb{N} \mid \forall m \geq n : \mathrm{sh}(w(m)) \geq \mathrm{sh}(w(n))\}$$

and $\mathsf{Stairs}(w) = w(n_0)w(n_1)\cdots$, where $n_0 < n_1 < \cdots$ is the ascending enumeration of $\mathsf{StairPositions}(w)$.

Now, using the notion of stairs, we define stair-score functions for pushdown games. To simplify our notation, let $\mathsf{reset}(v) = \varepsilon$ and $\mathsf{lastBump}(v) = v$ for $v \in V$ and for $w = w(0)\cdots w(r)$ with $r \geq 1$, let $\mathsf{reset}(w) = w(0)\cdots w(l)$ and $\mathsf{lastBump}(w) = w(l+1)\cdots w(r)$, where l is the greatest position such that $\mathrm{sh}(w(l)) \leq \mathrm{sh}(w(r))$ and $l \neq r$, i.e., l is the second largest[1] stair position of w. Figure 2 illustrates the above definitions, where an example path w and the corresponding stack heights are depicted. The stair positions are indicated by the marked stack heights. Furthermore, the figure also illustrates our new definition of stair-scores which we define next.

Definition 8 (Stair-scoring function)**.** *For every color $c \in [n]$, define the function $\mathsf{StairSc}_c\colon V^* \to \mathbb{N}$ by $\mathsf{StairSc}_c(\varepsilon) = 0$ and for $w \in V^+$ by*

$$\mathsf{StairSc}_c(w) = \begin{cases} \mathsf{StairSc}_c(\mathsf{reset}(w)) & \text{if } \mathsf{MinCol}(\mathsf{lastBump}(w)) > c, \\ \mathsf{StairSc}_c(\mathsf{reset}(w)) + 1 & \text{if } \mathsf{MinCol}(\mathsf{lastBump}(w)) = c, \\ 0 & \text{if } \mathsf{MinCol}(\mathsf{lastBump}(w)) < c. \end{cases}$$

[1] Notice that the last position of a finite path is always a stair position.

Furthermore, for every color $c \in [n]$*, the function* $\mathsf{MaxStairSc}_c \colon V^* \cup V^\omega \to \mathbb{N} \cup \{\infty\}$ *is defined by* $\mathsf{MaxStairSc}_c(\rho) = \max_{w \sqsubseteq \rho} \mathsf{StairSc}_c(w)$.

Now, using these notions we define finite-time pushdown games. Such a game (G, col, k) consists of a pushdown game graph G, a min-parity condition col and a threshold $k \in \mathbb{N} \setminus \{0\}$. A play in (G, col, k) is a finite path $w = w(0) \cdots w(r) \in V^*$ with $w(0) = v_{\mathrm{in}}$ such that $\mathsf{MaxStairSc}_c(w) = k$ for some $c \in [n]$, and $\mathsf{MaxStairSc}_c(w(0) \cdots w(r-1)) < k$ for all $c \in [n]$. The play w is winning for Player i if $\mathrm{Par}(c) = i$. Again, the notions of (winning) strategies are defined as usual.

As above, every threshold k is eventually reached by some stair-score function if the play is sufficiently long: a simple induction shows that every $w \in V^+$ with $0 \in \mathsf{StairPositions}(w)$ and with $|w| \geq 2^m$ has a prefix $w' \sqsubseteq w$ such that $|\mathsf{StairPositions}(w')| > m$. Furthermore, for every play prefix $w' \sqsubseteq w$ a sequence $u' \in Q^*$ of states with $|u'| = |\mathsf{Stairs}(w')|$ can be constructed such that for every color $c \in [n]$, $\mathsf{StairSc}_c(w') = \mathsf{Sc}_c(u')$. Combining these two properties and Lemma 4 yields the desired upper bound on the length of a play.

Lemma 9. *For every* $w \in V^*$ *with* $|w| \geq 2^{k^n}$ *there is some* $c \in [n]$ *such that* $\mathsf{MaxStairSc}_c(w) \geq k$.

Thus, a play in a finite-time pushdown game stops after a doubly-exponential number of rounds. Again, the bound in Lemma 9 is tight. Moreover, Lemma 5 can directly be translated to the new definition of stair-scoring functions which ensures a unique winner of a play.

Lemma 10. *Let* $w \in V^*$*,* $v \in V$ *and* $c, c' \in [n]$*. If* $\mathsf{StairSc}_c(wv) = \mathsf{StairSc}_c(w) + 1$ *and* $\mathsf{StairSc}_{c'}(wv) = \mathsf{StairSc}_{c'}(w) + 1$*, then* $c = c'$.

In Section 5, we prove the equivalence between pushdown games and finite-time pushdown games. To this end, we adapt Walukiewicz's reduction from pushdown parity games to parity games on finite game graphs, which we recall in the following section.

4 Walukiewicz's Reduction

Walukiewicz showed that pushdown games can be solved in exponential time [9]. In this section, we recall his technique which comprises a reduction to parity games on finite game graphs. We present a slight modification of the original construction which is needed to prove our result in the next section.

Let $\mathscr{G} = (G, \mathrm{col})$ be a pushdown game with game graph $G = (V, V_0, V_1, E, v_{\mathrm{in}})$ induced by $\mathscr{P} = (Q, \Gamma, \Delta, q_{\mathrm{in}})$ with partition $Q_0 \cup Q_1$ of Q and min-parity condition $\mathrm{col} \colon Q \to [n]$. To simulate $\mathscr{G}$ by a game on a finite game graph the information stored on the stack is encoded by some finite memory structure. The essential component of this structure is the set $\mathsf{Pred} = (\mathscr{P}(Q))^n$, which we call the set of predictions. A prediction $P = (P_0, \dots, P_{n-1}) \in \mathsf{Pred}$ contains for every $c \in [n]$ a subset $P_c \subseteq Q$ of states.

The core idea of the game simulating the pushdown game is the following: The players are assigned different tasks, one of them makes predictions and the other one verifies them. Whenever a push-transition is to be simulated the predicting player has to make a prediction $P \in \mathsf{Pred}$ about the future round t when the same stack height as before performing the push-transition is reached again for the first time (if it is reached at all). With this prediction, the predicting player claims that if the current push-transition is performed, then in round t some state $q \in P_c$ will be reached if $c \in [n]$ is the minimal color seen in between. Once a prediction P is proposed, the verifying player has two ways of reacting, either believing that P is correct or not. In the first case, he is not interested in verifying P, so the push-transition is not performed and the verifying player chooses a color $c \in [n]$ and a state $q \in P_c$, for some $P_c \neq \emptyset$, and skips a part of the simulated play by jumping to an appropriate position in the play. In the other case, he wants to verify the correctness of P, so the push-transition is performed and when the top of the stack is eventually popped it will turn out whether P is correct or not. The predicting player wins if P turns out to be correct and otherwise the verifying player wins. So after a pop-transition the winner is certain. For the other case, where no pop-transition is performed at all, the parity condition determines the winner.

In the following, let Player i take the role of the predicting player and Player $1-i$ the role of the verifying one. The game $\mathscr{G}'_i = (G', \mathrm{col}')$ which depends on $i \in \{0,1\}$, with $G' = (V', V'_0, V'_1, E', v'_{\mathrm{in}})$ is defined as follows: For all states $q \in Q$, stack symbols $A, B \in \Gamma_\perp$, colors $c, d \in [n]$ and predictions $P, R \in \mathsf{Pred}$, the set V' contains the vertices $\mathsf{Check}[q,A,P,c,d]$ which correspond to the configurations of $\mathscr{G}$, auxiliary vertices $\mathsf{Push}[P,c,q,AB]$, $\mathsf{Claim}[P,c,q,AB,R]$ and $\mathsf{Jump}[q,A,P,c,d]$ which serve as intermediates to signalize the intention to perform a push-transition, to make a new prediction and to skip a part of a simulated play, and finally the sink vertices $\mathsf{Win}_i[q]$ and $\mathsf{Win}_{1-i}[q]$.

The set E' consists of the following edges (for the sake of readability, we denote an edge $(v_1, v_2) \in E'$ here by $v_1 \to v_2$). For every skip-transition $\delta = (q,A,p,B) \in \Delta$ there are edges

$$\mathsf{Check}[q,A,P,c,d] \to \mathsf{Check}[p,B,P,\min\{c,\mathrm{col}(p)\},\mathrm{col}(p)]\,,$$

for $P \in \mathsf{Pred}$ and $c, d \in [n]$. Thus, the first two components of the Check-vertices are updated according to δ, the prediction P remains untouched, the last but one component is used to keep track of the minimal color for being able to check the prediction for correctness and the last component determines the color of the current Check-vertex. For every push-transition $\delta = (q,A,p,BC) \in \Delta$ there are edges

$$\mathsf{Check}[q,A,P,c,d] \to \mathsf{Push}[P,c,p,BC]\,,$$

for all $P \in \mathsf{Pred}$ and $c, d \in [n]$. Here, a player states that a push-transition is to be performed such that the current state q has to be changed to p and the top of the stack A has to be replaced by BC. The information containing the current prediction P and the minimal color c is carried over, as this is needed in the case where the verifying player decides to skip. Moreover, to make a new prediction R, all edges

$$\mathsf{Push}[P,c,p,BC] \to \mathsf{Claim}[P,c,p,BC,R]$$

for every $R \in \mathsf{Pred}$ are needed. In case a new prediction is to be verified, a push-transition is finally performed using edges of the form

$$\mathsf{Claim}[P,c,p,BC,R] \to \mathsf{Check}[p,B,R,\mathrm{col}(p),\mathrm{col}(p)]$$

where the prediction P, the color c and the lower stack symbol C are discarded, since they are no longer needed. For the other case, where the verifying player intends to skip a part of a play, all edges

$$\mathsf{Claim}[P,c,p,BC,R] \to \mathsf{Jump}[q,C,P,c,e]$$

with $q \in R_e$ are contained in E'. Here, the verifying player chooses a color $e \in [n]$ for the minimal color of the skipped part and a state q from the corresponding component R_e of the prediction R. Now, the lower stack symbol C, the prediction P and the color c additionally have to be carried over, whereas B and R are discarded. Then, all edges

$$\mathsf{Jump}[q,C,P,c,e] \to \mathsf{Check}[q,C,P,\min\{c,e,\mathrm{col}(q)\},\min\{e,\mathrm{col}(q)\}]$$

are contained in E' where the last component of the Check-vertex is set to be the minimum of the color of the current state q and the minimal color of the part just skipped. For the last but one component, we also have to account for the color c, which is necessary for eventually checking P for correctness. Finally, we have for every pop-transition $(q,A,p,\varepsilon) \in \Delta$, the edges

$$\mathsf{Check}[q,A,P,c,d] \to \mathsf{Win}_i[p] \quad \text{if } p \in P_c\,, \text{ and}$$
$$\mathsf{Check}[q,A,P,c,d] \to \mathsf{Win}_{1-i}[p] \quad \text{if } p \notin P_c\,,$$

for $P \in \mathsf{Pred}$ and $c,d \in [n]$, which lead to the sink vertex of the predicting player $\mathsf{Win}_i[p]$ if the prediction P turns out to be correct or to the sink vertex of the verifying player $\mathsf{Win}_{1-i}[p]$ otherwise. Moreover, we have $(\mathsf{Win}_j[q],\mathsf{Win}_j[q]) \in E'$, for $j \in \{0,1\}$ and $q \in Q$.

The initial vertex v'_{in} has to correspond to the initial configuration $v_{\text{in}} = (q_{\text{in}},\perp)$, so it is defined to be $\mathsf{Check}[q_{\text{in}},\perp,P^{\text{in}},\text{col}(q_{\text{in}}),\text{col}(q_{\text{in}})]$ where $P^{\text{in}}_c = \emptyset$ for every $c \in [n]$, as the $\perp$-symbol cannot be deleted from the stack. The set of vertices V'_i of the predicting Player i is defined to consist of all Push-vertices, as there Player i has to make a new prediction, and of those $\mathsf{Check}[p,A,P,c,d]$ vertices where $p \in Q_i$. Accordingly, all other vertices belong to Player $1-i$. Finally, the coloring function $\text{col}'\colon V' \to [n+1]$ is defined by $\text{col}'(\mathsf{Check}[p,A,P,c,d]) = d$ and $\text{col}'(\mathsf{Win}_j[q]) = j$, for $j \in \{0,1\}$. All other vertices are colored by the maximal color n (which does not appear in $\mathscr{G}$), since they are auxiliary vertices and should have no influence on the minimal color seen infinitely often. This is guaranteed by the structure of G', as there are no loops consisting only of auxiliary vertices. Notice that in the original construction, Jump-vertices are colored by the minimal color of the skipped part of the play which is chosen by the verifying player. This is avoided here by shifting the color of a Jump-vertex to the successive Check-vertex. For this purpose, the last component of the Check-vertices is introduced.

Theorem 11 ([9]). *Let $\mathscr{G}$ be a pushdown game. Player i wins $\mathscr{G}$ if and only if Player i wins $\mathscr{G}'_i$.*

Now, let us describe how a winning strategy σ for Player i in $\mathscr{G}$ can be constructed from a positional winning strategy σ'_i for Player i in $\mathscr{G}'_i$. The idea is to simulate σ'_i in $\mathscr{G}$. This works out fine as long as only skip- and push-transitions are involved. As soon as the first pop-transition is used, σ'_i leads to a sink Win_i-vertex at which the future moves of σ'_i are no longer useful for playing in the original game $\mathscr{G}$. To overcome this, the strategy σ uses a stack to store Claim-vertices visited during the simulated play. This allows us to reset the simulated play and to continue from the appropriate successor Jump-vertex of the Claim-vertex stored on the stack.

Formally, let $G'|_{\sigma'_i} = (V'|_{\sigma'_i},V'_0|_{\sigma'_i},V'_1|_{\sigma'_i},E'|_{\sigma'_i},v'_{\text{in}})$ be the game graph of $\mathscr{G}'_i$ restricted to the vertices and edges visited by σ'_i. This implies that every vertex from $V'_i|_{\sigma'_i}$ has a unique successor in $G'|_{\sigma'_i}$ and that Win_{1-i}-vertices are not contained in $V'_i|_{\sigma'_i}$. The pushdown transducer $\mathscr{T}_\sigma$ implementing σ is obtained from σ'_i by employing $G'|_{\sigma'_i}$ for its finite control and the Claim-vertices as its stack symbols.

The PDT implementing σ is defined by $\mathscr{T}_\sigma = (Q^\sigma,\Gamma^\sigma,\Delta^\sigma,q^\sigma_{\text{in}},\Sigma^\sigma_I,\Sigma^\sigma_O,\lambda^\sigma)$, where $Q^\sigma = V'|_{\sigma'_i}$, $\Gamma^\sigma = \{v \in V'|_{\sigma'_i} \mid v \text{ is a } \mathsf{Claim}\text{-vertex}\}$, $q^\sigma_{\text{in}} = v'_{\text{in}}$, $\Sigma^\sigma_I = \Sigma^\sigma_O = \Delta$. To define Δ^σ, we first define the labeling $\ell\colon E'|_{\sigma'_i} \to \Delta \cup \{\varepsilon\}$ which assigns to every edge in $E'|_{\sigma'_i}$ its corresponding transition $\delta \in \Delta$ by

$$\ell(v,v') = \begin{cases} (q,A,p,B) & \text{if } (v,v') = (\mathsf{Check}[q,A,P,c,d],\mathsf{Check}[p,B,P,c',d'])\,, \\ (q,A,p,BC) & \text{if } (v,v') = (\mathsf{Check}[q,A,P,c,d],\mathsf{Push}[P,c,p,BC])\,, \\ (q,A,p,\varepsilon) & \text{if } (v,v') = (\mathsf{Check}[q,A,P,c,d],\mathsf{Win}_i[p])\,, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Now, the transition relation Δ^σ is defined as follows: for every $(v,v') \in E'|_{\sigma'_i}$, if v is not a Claim-vertex *and* v' is not a Win_i-vertex, then $(v,Z,\ell(v,v'),v',Z) \in \Delta^\sigma$, for every $Z \in \Gamma^\sigma_\perp$. For the other cases, if v is a Claim-vertex and v' is a Check-vertex, then $(v,Z,\ell(v,v'),v',vZ) \in \Delta^\sigma$ for $Z \in \Gamma^\sigma_\perp$, i.e., the Claim-vertex v is pushed onto the stack. And finally, if $(v,v') = (\mathsf{Check}[q,A,P,c,d],\mathsf{Win}_i[p])$, then $(v,Z,\ell(v,v'),\mathsf{Jump}[p,C,R,e,c],\varepsilon) \in \Delta^\sigma$ for every $Z \in \Gamma^\sigma$ of the form $Z = \mathsf{Claim}[R,e,q',BC,R']$, i.e., the topmost symbol $\mathsf{Claim}[R,e,q',BC,R']$ is popped from the stack and the pushdown transducer proceeds to the state $\mathsf{Jump}[p,C,R,e,c]$ which would be reached in $G'_i|_{\sigma'_i}$ if Player $1-i$ would have chosen color c and state $p \in R_c$ to determine the successor of $\mathsf{Claim}[R,e,q',BC,R']$. To complete the definition of $\mathscr{T}_\sigma$, we define the output function λ^σ by $\lambda^\sigma(v) = \ell(v,v')$ if $v \in V'_i|_{\sigma'_i}$ is a Check-vertex and $(v,v') \in E'|_{\sigma'_i}$, i.e., the labeling of the edge chosen by σ'_i determines the output of $\mathscr{T}_\sigma$. Lemma 13 shows this construction to be correct.

5 Main Theorem

In this section, we prove the equivalence between a pushdown game and the corresponding finite-time pushdown game for a certain threshold which is exponential in the size of the PDS defining the pushdown game. For a pushdown game $\mathscr{G} = (G,\mathrm{col})$ induced by $\mathscr{P} = (Q,\Gamma,\Delta,q_{\mathrm{in}})$ and $\mathrm{col}\colon Q \to [n]$, define $k_{\mathscr{G}} = |Q|\cdot|\Gamma|\cdot 2^{|Q|\cdot n}\cdot n$, which is an upper bound on the number of Check-vertices in $\mathscr{G}'_i$ of the same color.

Theorem 12. *Let $\mathscr{G} = (G,\mathrm{col})$ be a pushdown game and let $\mathscr{G}_k = (G,\mathrm{col},k)$ be the corresponding finite-time pushdown game with threshold k. For every $k > k_{\mathscr{G}}$, Player i wins $\mathscr{G}$ if and only if Player i wins $\mathscr{G}_k$.*

To prove this theorem, we need the following lemma which establishes a relation between the values of the scoring functions of plays in $\mathscr{G}'_i$ and the values of the stair-scoring functions of corresponding plays in $\mathscr{G}$. Let σ'_i be a positional winning strategy for Player i in $\mathscr{G}'_i$ and $\mathscr{T}_\sigma$ the PDT implementing the corresponding pushdown winning strategy σ for Player i in $\mathscr{G}$ as defined in the previous section. For a play prefix $w(0)\cdots w(r) \in V^+$, define $\mathsf{lastStrictBump}(w) = w$ if $\mathrm{sh}(w(r)) = 0$, and otherwise $\mathsf{lastStrictBump}(w) = w(l+1)\cdots w(r)$ where l is the greatest position such that $\mathrm{sh}(w(l)) < \mathrm{sh}(w(r))$.

Lemma 13. *For every play prefix w in $\mathscr{G}$ that is consistent with σ, there is a play prefix w' in $\mathscr{G}'_i$ that is consistent with σ'_i such that $\mathsf{StairSc}_c(w) = \mathsf{Sc}_c(w')$ for every $c \in [n]$.*

Proof. By induction over $|w|$. To prove our claim, we strengthen the induction hypothesis as follows: for every play prefix w in $\mathscr{G}$ that is consistent with σ, there is a play prefix w' in $\mathscr{G}'_i|_{\sigma'_i}$ (which is consistent with σ'_i by construction) such that the following requirements are satisfied: let $\mathsf{last}(w) = (q,A\gamma)$.

(i) $\mathsf{StairSc}_c(w) = \mathsf{Sc}_c(w')$ for every $c \in [n]$.

(ii) $\mathsf{last}(w') = \mathsf{Check}[q,A,P,c,d]$ for some $P \in \mathsf{Pred}$, $d \in [n]$ and $c = \mathsf{MinCol}(\mathsf{lastStrictBump}(w))$.

(iii) Let (v,γ_σ) be the last configuration of the run of $\mathscr{T}_\sigma$ on the sequence of transitions induced by w. Furthermore, if $\gamma_\sigma \neq \perp$, let $\gamma_\sigma(j) = \mathsf{Claim}[P_j,c_j,p_j,B_jC_j,R_j]$ for every $0 \le j \le |\gamma_\sigma|-2$. We require $v = \mathsf{last}(w')$, $C_0\cdots C_k = \gamma$ where $k = |\gamma_\sigma|-2$, and $R_0 = P$.

For the induction start, we have $w = v_{\mathrm{in}} = (q_{\mathrm{in}},\perp)$. Let $w' = v'_{\mathrm{in}} = \mathsf{Check}[q_{\mathrm{in}},\perp,P^{\mathrm{in}},\mathrm{col}(q_{\mathrm{in}}),\mathrm{col}(q_{\mathrm{in}})]$. Since $\mathrm{col}(v_{\mathrm{in}}) = \mathrm{col}'(v'_{\mathrm{in}}) = \mathrm{col}(q_{\mathrm{in}})$, we have $\mathsf{StairSc}_c(w) = \mathsf{Sc}_c(w')$ for every $c \in [n]$. Moreover, requirements (ii) and (iii) are satisfied as well.

Now, let $w = w(0)\cdots w(r)$ with $r > 0$ and $w(r-1) = (q,A\gamma)$. Moreover, let $\mathsf{reset}(w) = w(0)\cdots w(s)$ and $w(s) = (q_s,A_s\gamma_s)$. The induction hypothesis yields play prefixes u' and u'_s in $\mathscr{G}'_i|_{\sigma'_i}$ such that we have $\mathsf{StairSc}_c(w(0)\cdots w(r-1)) = \mathsf{Sc}_c(u')$ and $\mathsf{StairSc}_c(w(0)\cdots w(s)) = \mathsf{Sc}_c(u'_s)$, for every $c \in [n]$. Also, for some $P,P_s \in \mathsf{Pred}$ and $d,d_s \in [n]$, $\mathsf{last}(u') = \mathsf{Check}[q,A,P,c,d]$ and $\mathsf{last}(u'_s) = \mathsf{Check}[q_s,A_s,P_s,c_s,d_s]$ with $c = \mathsf{MinCol}(\mathsf{lastStrictBump}(w(0)\cdots w(r-1)))$ and $c_s = \mathsf{MinCol}(\mathsf{lastStrictBump}(w(0)\cdots w(s)))$. We distinguish three cases, whether the transition from $w(r-1)$ to $w(r)$ is a skip-, push-, or pop-transition.

In case of a skip-transition $\delta = (q,A,p,B)$, we have $w(r) = (p,B\gamma)$. By construction, there is also an edge from $\mathsf{last}(u') = \mathsf{Check}[q,A,P,c,d]$ to the vertex

$$v = \mathsf{Check}[p,B,P,\min\{c,\mathrm{col}(p)\},\mathrm{col}(p)]$$

in $\mathscr{G}_i'|_{\sigma_i'}$ labeled by $\ell(\mathsf{last}(u'),v) = \delta$. Thus, let $w' = u'v$. This choice satisfies requirement (ii), as for a skip-transition from $w(r-1)$ to $w(r)$ it holds

$$\begin{aligned}\mathsf{MinCol}(\mathsf{lastStrictBump}(w)) &= \min\{\mathsf{MinCol}(\mathsf{lastStrictBump}(w(0)\cdots w(r-1))),\mathrm{col}(w(r))\}\\ &= \min\{c,\mathrm{col}(p)\}\ .\end{aligned}$$

Furthermore, requirement (iii) is satisfied, since when processing δ, $\mathscr{T}_\sigma$ changes its state $\mathsf{last}(u')$ to v while the stack is left unchanged. To prove the equality of the scores, let $e = \mathrm{col}(w(r))$, which is also the color of v in $\mathscr{G}_i'|_{\sigma_i'}$. Then, we have $\mathsf{StairSc}_e(w) = \mathsf{StairSc}_e(w(0)\cdots w(r-1))+1 = \mathsf{Sc}_e(u')+1 = \mathsf{Sc}_e(w')$, and for $e' < e$, $\mathsf{StairSc}_{e'}(w) = \mathsf{StairSc}_{e'}(w(0)\cdots w(r-1)) = \mathsf{Sc}_{e'}(u') = \mathsf{Sc}_{e'}(w')$. Finally, for $e' > e$, we have $\mathsf{StairSc}_{e'}(w) = 0 = \mathsf{Sc}_{e'}(w')$.

In case of a push-transition $\delta = (q,A,p,BC)$, we have $w(r) = (p,BC\gamma)$. Consider the finite path

$$u'' = \mathsf{Push}[P,c,p,BC] \to \mathsf{Claim}[P,c,p,BC,R] \to \mathsf{Check}[p,B,R,\mathrm{col}(p),\mathrm{col}(p)]$$

in $\mathscr{G}_i'|_{\sigma_i'}$ where R is the prediction picked by σ_i'. Notice that there is indeed an edge from $\mathsf{last}(u')$ to $\mathsf{Push}[P,c,p,BC]$ in $E'|_{\sigma_i'}$. We claim that $w' = u'u''$ has the desired properties. Requirement (ii) is satisfied, as $\mathsf{lastStrictBump}(w) = w(r)$ in this case, and $\mathsf{MinCol}(w(r)) = \mathrm{col}(p)$. Furthermore, $\mathsf{Claim}[P,c,p,BC,R]$ is pushed onto the stack of $\mathscr{T}_\sigma$ when processing δ. Hence, requirement (iii) is satisfied.

The scores evolve as in the case of a skip-transition explained above, since in both cases we have $\mathsf{lastBump}(w) = w(r)$, and u'' contains exactly one vertex with color in $[n]$, namely its last vertex, which has the same color as $w(r)$. The intermediate auxiliary vertices have color n and therefore do not influence the scores we are interested in.

Finally, the case of a pop-transition is the most involved one, since a play in $\mathscr{G}_i'|_{\sigma_i'}$ ends in a sink vertex, as soon as a pop-transition is simulated. In this case, $\mathscr{T}_\sigma$ uses the top Claim-vertex stored on its stack to determine the appropriate Check-vertex for being able to continue playing according to σ_i'. Suppose the transition is $\delta = (q,A,p,\varepsilon)$, i.e., we have $w(r) = (p,\gamma)$. Let $\delta_s = (q_s,A_s,q',BC)$ be the push-transition (of the PDS underlying $\mathscr{G}$) which induces the edge $(w(s),w(s+1)) \in E$. Note that $C\gamma_s = \gamma$, since the stack content $C\gamma_s$ remains untouched until δ is executed from $w(r-1)$ to $w(r)$. Hence, $w(r) = (p,C\gamma_s)$. By definition of σ, there is an edge from $\mathsf{last}(u') = \mathsf{Check}[q,A,P,c,d]$ to $\mathsf{Win}_i[p]$ in $E'|_{\sigma_i'}$ such that $p \in P_c$.

Now, consider the run of $\mathscr{T}_\sigma$ on w. By construction, the transducer pops the top Claim-vertex v from its stack while processing the transition δ. We show that $v = \mathsf{Claim}[P_s,c_s,q',BC,P]$. First, notice that v was pushed onto the stack while processing the transition from $w(s)$ to $w(s+1)$ which is induced by δ_s. Applying the induction hypothesis shows that the run of $\mathscr{T}_\sigma$ on the sequence of transitions induced by $w(0)\cdots w(s)$ ends in state $\mathsf{last}(u_s') = \mathsf{Check}[q_s,A_s,P_s,c_s,d_s]$ with some stack content $\gamma_\sigma \in (\Gamma^\sigma)^+\bot$ satisfying the above requirements. Since now δ_s is to be processed, the run of $\mathscr{T}_\sigma$ is continued as follows for some $R \in \mathsf{Pred}$:

$$\begin{aligned}(\mathsf{last}(u_s'),\gamma_\sigma) &\overset{\delta_s}{\vdash} (\mathsf{Push}[P_s,c_s,q',BC],\gamma_\sigma) \overset{\varepsilon}{\vdash} (\mathsf{Claim}[P_s,c_s,q',BC,R],\gamma_\sigma)\\ &\overset{\varepsilon}{\vdash} (\mathsf{Check}[q',B,R,\mathrm{col}(q'),\mathrm{col}(q')],\mathsf{Claim}[P_s,c_s,q',BC,R]\cdot\gamma_\sigma)\end{aligned}$$

It remains to show that $R = P$, which is done by applying the induction hypothesis to the run of $\mathscr{T}_\sigma$ on transitions induced by $w(0)\cdots w(r-1)$. The top symbol $\mathsf{Claim}[P_s, c_s, q', BC, R]$, which is pushed on the stack while processing $(w(s), w(s+1))$, remains untouched until $w(r-1)$ is reached and is again the top symbol after processing $(w(r-2), w(r-1))$. However, since $\mathsf{last}(u') = \mathsf{Check}[q, A, P, c, d]$ is the state reached by $\mathscr{T}_\sigma$ after processing $w(0)\cdots w(r-1)$ it follows from requirement (iii) that $R = P$.

Consider the following finite path in $\mathscr{G}'_i|_{\sigma'_i}$:

$$u'' = \mathsf{Push}[P_s, c_s, q', BC] \to v \to \mathsf{Jump}[p, C, P_s, c_s, c] \to \mathsf{Check}[p, C, P_s, \mathsf{min}\{c_s, c, \mathrm{col}(p)\}, \mathsf{min}\{c, \mathrm{col}(p)\}].$$

Notice that there is an edge from $\mathsf{last}(u'_s)$ to $\mathsf{Push}[P_s, c_s, q', BC]$ in $E'|_{\sigma'_i}$. So, we can show that $w' = u'_s u''$ satisfies the above requirements. Requirement (ii) is satisfied, since

$$\begin{aligned}
&\mathsf{MinCol}(\mathsf{lastStrictBump}(w))\\
=&\,\mathsf{min}\{\mathsf{MinCol}(\mathsf{lastStrictBump}(w(0)\cdots w(s))), \mathsf{MinCol}(w(s+1)\cdots w(r-1)), \mathrm{col}(w(r))\}\\
=&\,\mathsf{min}\{c_s, \mathsf{MinCol}(\mathsf{lastStrictBump}(w(0)\cdots w(r-1))), \mathrm{col}(p)\}\\
=&\,\mathsf{min}\{c_s, c, \mathrm{col}(p)\} \quad .
\end{aligned}$$

Requirement (iii) is satisfied, since after processing δ by $\mathscr{T}_\sigma$, the top stack symbol v is popped from the stack and the state $\mathsf{Check}[p, C, P_s, \mathsf{min}\{c_s, c, \mathrm{col}(p)\}, \mathsf{min}\{c, \mathrm{col}(p)\}]$ is reached. By doing so, the same stack content is reestablished as after the run of $\mathscr{T}_\sigma$ on $\mathsf{reset}(w)$. Hence, by applying the induction hypothesis, we have $C_0 \cdots C_k = \gamma_s$. Since we have $\gamma = C\gamma_s$, this suffices. To show requirement (i), let

$$\begin{aligned}
e &= \mathsf{MinCol}(\mathsf{lastBump}(w))\\
&= \mathsf{min}\{\mathsf{MinCol}(\mathsf{lastStrictBump}(w(0)\cdots w(r-1))), \mathrm{col}(w(r))\}\\
&= \mathsf{min}\{c, \mathrm{col}(p)\} \quad .
\end{aligned}$$

Notice that e is also the color of $\mathsf{last}(w') = \mathsf{Check}[p, C, R, \mathsf{min}\{c_s, c, \mathrm{col}(p)\}, \mathsf{min}\{c, \mathrm{col}(p)\}]$ in $\mathscr{G}'_i|_{\sigma'_i}$. Thus, $\mathsf{StairSc}_e(w) = \mathsf{StairSc}_e(w(0)\cdots w(s)) + 1 = \mathsf{Sc}_e(u'_s) + 1 = \mathsf{Sc}_e(w')$ and for $e' < e$ $\mathsf{StairSc}_{e'}(w) = \mathsf{StairSc}_{e'}(w(0)\cdots w(s)) = \mathsf{Sc}_{e'}(u'_s) = \mathsf{Sc}_{e'}(w')$. Finally, if $e' > e$, $\mathsf{StairSc}_{e'}(w) = 0 = \mathsf{Sc}_{e'}(w')$. □

Now, the proof of Theorem 12 is straightforward.

Proof of Theorem 12. Assume that Player i wins $\mathscr{G}$, then he also wins $\mathscr{G}'_i$ due to Theorem 11. For every color $c \in [n]$, there are at most $k_{\mathscr{G}}$ Check-vertices colored by c. Hence, due to Remark 3 there is a positional winning strategy σ'_i in $\mathscr{G}'_i$ for Player i such that for every $c \in [n]$ with $\mathrm{Par}(c) = 1 - i$, $\mathsf{MaxSc}_c(\rho') \leq k_{\mathscr{G}}$, for every play ρ' which is consistent with σ'_i. From Lemma 13, it follows that the pushdown strategy σ which is constructed from σ'_i bounds the stair-scores of Player $1-i$ by $k_{\mathscr{G}}$. Thus, for every play ρ which is consistent with σ and every $k > k_{\mathscr{G}}$, there exists $w \sqsubset \rho$ such that w is winning for Player i in $\mathscr{G}_k$. Hence, using the same strategy σ Player i wins every finite-time game $\mathscr{G}_k$ for $k > k_{\mathscr{G}}$. The other direction follows by determinacy of parity games. □

6 Lower Bounds

In the previous section, we proved the equivalence between pushdown games and corresponding finite-time pushdown games with an exponential threshold. In this section, we present an (almost) matching lower bound on the threshold that always yields equivalent games. To this end, we construct a pushdown

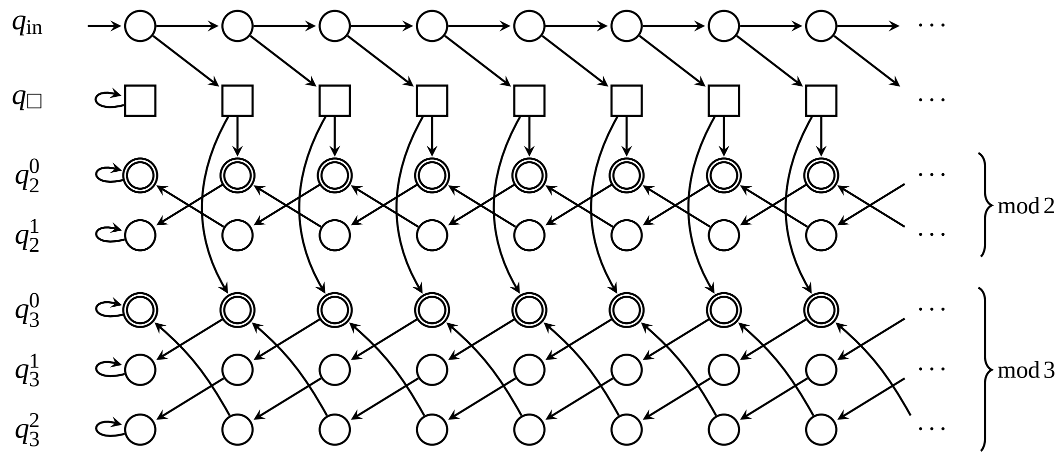


Figure 3: Pushdown Game (G_2, col_2)

game in which the winning player is forced to reach a configuration of *high* stack height while only visiting states colored by a bad color for him. Thereby, the opponent is the first player to reach *high* stair-scores, although he loses the play eventually.

Theorem 14. *There are a family of pushdown games (G_n, col_n) and thresholds k_n exponential in the cubic root of the size of the underlying PDS such that for every $n > 0$, Player 0 wins the pushdown game (G_n, col_n), but for every $k \leq k_n$, Player 1 wins the finite-time pushdown game (G_n, col_n, k).*

Proof. We denote the i-th prime number by p_i. For $n > 0$, let $k_n = \prod_{i=1}^n p_i$ and define the PDS $\mathscr{P}_n = (Q_n, \{A\}, \Delta_n, q_{\text{in}})$ as follows: $Q_n = \{q_{\text{in}}, q_{\Box}\} \cup \bigcup_{i=1}^n M_i$, where $M_i = \{q_i^j \mid 0 \leq j < p_i\}$, and Δ_n consists of the following transitions:

- $(q_{\text{in}}, X, q_{\text{in}}, AX)$ and $(q_{\text{in}}, X, q_{\Box}, AX)$ for every $X \in \{A, \perp\}$,
- $(q_{\Box}, A, q_i^0, A)$ for every $1 \leq i \leq n$,
- $(q_i^j, A, q_i^\ell, \varepsilon)$, where $\ell = (j+1) \bmod p_i$, and
- $(q, \perp, q, \perp)$, for every $q \in Q_n \setminus \{q_{\text{in}}\}$.

To specify the partition of Q_n, let $q_{\Box}$ belong to Player 1. All other states are Player 0 states. The coloring is given by $\text{col}_n(q_i^0) = 0$ for every $1 \leq i \leq n$ and $\text{col}_n(q) = 1$ for every other state q. We have $k_n \geq 2^n$ and $|Q_n|$ can be bounded from above by $\mathscr{O}(n^2 \log(n))$. Hence, k_n is exponential in the cubic root of $|Q_n|$. The pushdown game (G_2, col_2) is depicted in Figure 3. Double-lined vertices are those colored by 0.

A play in the game (G_n, col_n) proceeds as follows. Player 0 picks a natural number $x > 0$ by moving the token to the configuration $(q_{\Box}, A^x\perp)$. If he fails to do so by staying in state q_{in} ad infinitum he loses, since $\text{col}_n(q_{\text{in}}) = 1$. At $(q_{\Box}, A^x\perp)$, Player 1 picks a modulus $p_i \in \{p_1, \ldots, p_n\}$ by moving the token to $(q_i^0, A^x\perp)$. From this configuration, a single path emanates, i.e., there is only one way to continue the play. Player 0 wins this play if and only if $x \bmod p_i = 0$. Hence, Player 0 has a winning strategy for this game by moving the token to some non-zero multiple of k_n, i.e., Player 0 wins (G_n, col_n).

Now, let $k \leq k_n$. If Player 0 reaches $(q_{\text{in}}, A^{k-1}\perp)$, then he loses the finite-time pushdown game (G_n, col_n, k), since in this case Player 1 reaches stair-score k for color 1. On the other hand, if he moves the token to a configuration $(q_{\Box}, A^x\perp)$ for some $x \leq k-1$, then there is a $p_i \in \{p_1, \ldots, p_n\}$ such that $x \bmod p_i \neq 0$, as $x < k_n$. Hence, assume Player 1 moves the token to $(q_i^0, A^x\perp)$. Then, the play ends in a self-loop at a configuration $(q_i^m, \perp)$ for some $m \neq 0$. The path w from $(q_{\mathrm{in}}, \perp)$ to $(q_i^m, \perp)$ via $(q_\Box, A^x)$ satisfies $\mathsf{MaxStairSc}_0(w) \leq x$. Since q_i^m is colored by 1, the scores of Player 0 are never increased while using the self-loop at $(q_i^m, \perp)$. Thus, his scores never reach the threshold k. Hence, Player 1 is the first to reach this threshold, since Lemma 9 guarantees that there is some color that reaches the threshold eventually. Thus, Player 1 wins (G_n, col_n, k). □

7 Conclusion

We have shown how to play parity games on pushdown graphs in finite time. To this end, we adapted the notions of scoring functions to exploit the intrinsic structure of a pushdown game graph to obtain an finite-duration game that always has the same winner as the infinite game. Thus, the winner of a parity game on a pushdown game graph can be determined by solving a finite reachability game.

This work transfers results obtained for games on finite game graphs to infinite graphs. In ongoing work, we investigate if and how a winning strategy for the safety game, in which Player 0 wins if and only if he prevents his opponent from reaching an exponential stair-score can be turned into a winning strategy for the original pushdown game. The winner of these two games is equal, due to Lemma 13.

On the other hand, our results could be extended by considering more general classes of infinite graphs having an intrinsic structure, e.g., configuration graphs of higher-order pushdown systems. Finally, there is a small gap between the upper and lower bound on the threshold score that always yields an equivalent finite-duration pushdown game, which remains to be closed.

References

[1] Julien Bernet, David Janin & Igor Walukiewicz (2002): *Permissive strategies: from parity games to safety games*. *ITA* 36(3), pp. 261–275, doi:10.1051/ita:2002013.

[2] E. Allen Emerson & Charanjit S. Jutla (1991): *Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)*. In: *FOCS*, IEEE Computer Society, pp. 368–377, doi:10.1109/SFCS.1991.185392.

[3] John Fearnley & Martin Zimmermann (2012): *Playing Muller Games in a Hurry*. *Int. J. Found. Comput. Sci.* 23(3), pp. 649–668, doi:10.1142/S0129054112400321.

[4] Marcin Jurdziński (2000): *Small Progress Measures for Solving Parity Games*. In Horst Reichel & Sophie Tison, editors: *STACS*, *Lecture Notes in Computer Science* 1770, Springer, pp. 290–301, doi:10.1007/3-540-46541-3_24.

[5] Christof Löding, P. Madhusudan & Olivier Serre (2004): *Visibly Pushdown Games*. In Kamal Lodaya & Meena Mahajan, editors: *FSTTCS*, *Lecture Notes in Computer Science* 3328, Springer, pp. 408–420, doi:10.1007/978-3-540-30538-5_34.

[6] Robert McNaughton (2000): *Playing Infinite Games in Finite Time*. In Arto Salomaa, Derick Wood & Sheng Yu, editors: *A Half-Century of Automata Theory*, World Scientific, pp. 73–91.

[7] Andrzej W. Mostowski (1991): *Games with forbidden positions*. Technical Report 78, University of Gdansk.

[8] Daniel Neider, Roman Rabinovich & Martin Zimmermann (2012): *Down the Borel-Hierarchy: Solving Muller Games via Safety Games*. In Marco Faella & Aniello Murano, editors: *GandALF 2012*, EPTCS. To appear.

[9] Igor Walukiewicz (1996): *Pushdown Processes: Games and Model Checking*. In Rajeev Alur & Thomas A. Henzinger, editors: *CAV*, *Lecture Notes in Computer Science* 1102, Springer, pp. 62–74, doi:10.1007/3-540-61474-5_58.