Infinite-state games with finitary conditions *

Krishnendu Chatterjee[1] and Nathanaël Fijalkow[2]

1 **IST Austria, Klosterneuburg, Austria**
`krishnendu.chatterjee@ist.ac.at`
2 **LIAFA, Université Denis Diderot-Paris 7, France**
Institute of Informatics, University of Warsaw, Poland
`nath@mimuw.edu.pl`

Abstract

We study two-player zero-sum games over infinite-state graphs equipped with ωB and finitary conditions.

Our first contribution is about the strategy complexity, *i.e* the memory required for winning strategies: we prove that over general infinite-state graphs, memoryless strategies are sufficient for finitary Büchi, and finite-memory suffices for finitary parity games.

We then study pushdown games with boundedness conditions, with two contributions. First we prove a collapse result for pushdown games with ωB-conditions, implying the decidability of solving these games. Second we consider pushdown games with finitary parity along with stack boundedness conditions, and show that solving these games is EXPTIME-complete.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Two-player games, Infinite-state systems, Pushdown games, Bounds in omega-regularity, Synthesis.

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Games on graphs. Two-player games played on graphs is a powerful mathematical framework to analyze several problems in computer science as well as mathematics. In particular, when the vertices of the graph represent the states of a reactive system and the edges represent the transitions, then the synthesis problem (Church's problem) asks for the construction of a winning strategy in a game played on the graph [12, 30]. Game-theoretic formulations have also proved useful for the verification, refinement, and compatibility checking of reactive systems [4]; and has deep connection with automata theory and logic, *e.g* the celebrated decidability result of monadic second-order logic over infinite trees due to Rabin [33].

Omega-regular conditions: strengths and weaknesses. In the literature, two-player games on finite-state graphs with ω-regular conditions have been extensively studied [21, 22, 25, 26]. The class of ω-regular languages provides a robust specification language for solving control and verification problems (see, *e.g*, [32]). Every ω-regular condition can be decomposed into a safety part and a liveness part [2]. The safety part ensures that the component will not do anything "bad" (such as violate an invariant) within any finite number of transitions. The liveness part ensures that the component will do something "good" (such

* The first author was supported by Austrian Science Fund (FWF) Grant No P23499 - N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), and Microsoft faculty fellows award. The second author has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 259454 (GALE) and n° 239850 (SOSNA).

as proceed, or respond, or terminate) in the long-run. Liveness can be violated only in the limit, by infinite sequences of transitions, as no bound is stipulated on when the "good" thing must happen. This infinitary, classical formulation of liveness has both strengths and weaknesses. A main strength is robustness, in particular, independence from the chosen granularity of transitions. Another important strength is simplicity, allowing liveness to serve as an abstraction for complicated safety conditions. For example, a component may always respond in a number of transitions that depends, in some complicated manner, on the exact size of the stimulus. Yet for correctness, we may be interested only that the component will respond "eventually". However, these strengths also point to a weakness of the classical definition of liveness: it can be satisfied by components that in practice are quite unsatisfactory because no bound can be put on their response time.

Stronger notion of liveness: finitary conditions. For the weakness of the infinitary formulation of liveness, alternative and stronger formulations of liveness have been proposed. One of these is *finitary* liveness [3]: it is satisfied if *there exists* a bound N such that every stimulus is followed by a response within N transitions. Note that it does not insist on a response within a known bound N (*i.e*, every stimulus is followed by a response within N transitions), but on response within some unknown bound, which can be arbitrarily large; in other words, the response time must not grow forever from one stimulus to the next. In this way, finitary liveness still maintains the robustness (independence of step granularity) and simplicity (abstraction of complicated safety conditions) of traditional liveness, while removing unsatisfactory implementations.

All ω-regular languages can be defined by a deterministic parity automaton; the parity condition assigns to each state an integer representing a priority, and requires that in the limit, every odd priority is followed by a lower even priority. Its finitary counterpart, the finitary parity condition, strenghtens this by requiring the existence of a bound N such that in the limit every odd priority is followed by a lower even priority within N transitions.

Games with finitary conditions. Games over finite graphs with finitary conditions have been studied in [15], leading to very efficient algorithms: finitary parity games can be solved in polynomial time. In this paper, we study games over infinite graphs with finitary conditions, and then focus on the widely studied class of pushdown games, which model sequential programs with recursion. This line of work belongs to the tradition of infinite-state systems and games, (see *e.g* [1, 11]). Pushdown games with the classical reachability and parity conditions have been studied in [5, 37]. It has been established in [37] that the problem of deciding the winner in pushdown parity games is EXPTIME-complete. However, little is known about pushdown games with boundedness conditions; one notable exception is parity and stack boundedness conditions [10, 24]. The stack boundedness condition naturally arises with the synthesis problem in mind, since bounding the stack amounts to control the depth of recursion calls of the sequential program.

Bounds in ω-regularity. The finitary conditions are closely related to the line of work initiated by Bojańczyk in [7], where the MSO $+\,\mathbb{B}$ logic was defined, generalizing MSO over infinite words by adding a bounding quantifier $\mathbb{B}$. The satisfiability problem for this logic has been deeply investigated (see for instance [7, 8, 9]), but the decidability for the general case is still open. A fragment of MSO $+\,\mathbb{B}$ over infinite words was shown to be decidable in [8], by introducing the model of ωB-automata, which manipulate counters. They perform three kind of actions on counters: increment (i), reset (r) or nothing (ε). The relation with finitary conditions has been investigated in [13], where it is shown that automata with finitary conditions exactly correspond to star-free ωB-expressions.

Regular cost-functions. A different perspective for bounds in ω-regularity was developed

by Colcombet in [16] with functions instead of languages, giving rise to the theory of regular cost-functions and cost-MSO. The decidability of cost-MSO over finite trees was established in [20], but its extension over infinite trees is still open, and would imply the decidability of the index of the non-deterministic Mostowski hierarchy [19], a problem open for decades. A subclass of cost-MSO called temporal cost logic was introduced in [18] and is the counterpart of finitary conditions for regular cost-functions [13].

Quantification order. The essential difference between the approaches underlying the logics MSO $+ \mathbb{B}$ and cost-MSO is a quantifier switch. We illustrate this in the context of games: a typical property expressed in MSO $+ \mathbb{B}$ is "there exists a strategy, such that for all plays, there exists a bound on the counter values", while cost-MSO allows to express properties like "there exists a strategy, there exists a bound N, such that for all plays, the counter values are bounded by N". In other words, MSO $+ \mathbb{B}$ expresses non-uniform bounds while bounds in cost-MSO are uniform.

Memoryless determinacy for infinite-state games. Colcombet pointed out in [17] that the remaining difficulty to establish the decidability of cost-MSO over infinite trees is a good understanding of cost-games, and more specifically the cornerstone is to extend the memoryless determinacy of parity games over infinite arenas, following [21, 22, 26].

This paper investigates the subcase of finitary conditions.

Our contributions. We study two questions about infinite-state games with finitary conditions: the *memory requirements* of winning strategies and the *decidability* of solving a pushdown game.

Strategy complexity. We give (non-effective) characterizations of the winning regions for finitary games over countably infinite graphs, implying a complete picture of the strategy complexity. Most importantly, we show that for finitary Büchi conditions memoryless strategies suffice, and that for finitary parity conditions, memory of size $d/2 + 1$ suffices, where d is the number of priorities of the parity condition.

Pushdown games. We present two contributions.

First we consider pushdown games with ωB-conditions and prove the equivalence between the following: "there exists a strategy, such that for all plays, there exists a bound on the counter values" and "there exists a strategy, there exists a bound N, such that for all plays, *eventually* the counter values are bounded by N". We refer to this as a collapse result, as it reduces a quantification with non-uniform bounds (in the fashion of MSO $+ \mathbb{B}$) to one with uniform bounds (à la cost-MSO). Using this, we obtain the decidability of determining the winner in such games relying on previous results [6, 7].

Second we consider pushdown games with finitary parity along with stack boundedness conditions, and establish that solving these games is EXPTIME-complete.

All proofs are omitted, but can be found in the technical report [14].

2 Definitions

Arenas and games. The games we consider are played on an *arena* $\mathcal{A} = (V, (V_E, V_A), E)$, which consists of a (potentially infinite but countable) graph (V, E) and a partition (V_E, V_A) of the vertex set V. A vertex is controlled by Eve and depicted by a circle if it belongs to V_E and controlled by Adam and depicted by a square if it belongs to V_A. Playing consists in moving a pebble along the edges: initially placed on a vertex v_0, the pebble is sent along an edge chosen by the player who controls the vertex. From this infinite interaction results a *play* π, which is an infinite sequence of vertices $v_0, v_1, \ldots$ where for all i, we have $(v_i, v_{i+1}) \in E$, *i.e* π is an infinite path in the graph. We denote by Π the set of all plays,

and define *conditions* for a player by sets of winning plays $\Omega \subseteq \Pi$. The games are zero-sum, which means that if Eve's condition is Ω, then Adam's condition is $\Pi \setminus \Omega$, usually denoted by "CoΩ" (the conditions are opposite). Formally, a *game* is given by $\mathcal{G} = (\mathcal{A}, \Omega)$ where $\mathcal{A}$ is an arena and Ω a condition. A condition Ω is prefix-independent if it is closed under adding and removing prefixes.

Strategies. A *strategy* for a player is a function that prescribes, given a finite history of the play, the next move. Formally, a *strategy* for Eve is a function $\sigma : V^* \cdot V_E \to V$ such that for a finite history $w \in V^*$ and a current vertex $v \in V_E$, the prescribed move is legal, *i.e* along an edge: $(v, \sigma(w \cdot v)) \in E$. Strategies for Adam are defined similarly, and usually denoted by τ. Once a game $\mathcal{G} = (\mathcal{A}, \Omega)$, a starting vertex v_0 and strategies σ for Eve and τ for Adam are fixed, there is a unique play denoted by $\pi(v_0, \sigma, \tau)$, which is said to be winning for Eve if it belongs to Ω. The sentence "Eve wins from v_0" means that she has a winning strategy from v_0, that is a strategy σ such that for all strategies τ for Adam, the play $\pi(v_0, \sigma, \tau)$ is winning. By "solving the game", we mean (algorithmically) determine the winner. We denote by $\mathcal{W}_E(\mathcal{G})$ the set of vertices from which Eve wins, also referred to as winning set, or winning region, and analogously $\mathcal{W}_A(\mathcal{G})$ for Adam. A very important theorem in game theory, due to Martin [29], states that Borel games (that is, where the condition is Borel) are determined, *i.e* we have $\mathcal{W}_E(\mathcal{G}) \cup \mathcal{W}_A(\mathcal{G}) = V$, *i.e* from any vertex, exactly one of the two players has a winning strategy. Throughout this paper, we only consider Borel conditions, hence our games are determined.

Memory structures. We define memory structures and strategies relying on memory structures. A *memory structure* $\mathcal{M} = (M, m_0, \mu)$ for an arena $\mathcal{A}$ consists of a set M of memory states, an initial memory state $m_0 \in M$, and an update function $\mu : M \times E \to M$. A memory structure is similar to an automaton synchronized with the arena: it starts from m_0 and reads the sequence of edges produced by the arena. Whenever an edge is taken, the current memory state is updated using the update function μ. A strategy relying on a memory structure $\mathcal{M}$, whenever it picks the next move, considers only the current vertex and the current memory state: it is thus given by a next-move function $\nu : V_E \times M \to V$. Formally, given a memory structure $\mathcal{M}$ and a next-move function ν, we can define a strategy σ for Eve by $\sigma(w \cdot v) = \nu(v, \mu^*(w \cdot v))$, where μ is extended to $\mu^* : V^+ \to M$. A strategy with memory structure $\mathcal{M}$ has finite memory if M is a finite set. It is *memoryless*, or *positional* if M is a singleton: in this case, the choice for the next move only depends on the current vertex, and can be described as a function $\sigma : V_E \to V$.

Attractors. Given $F \subseteq V$, define $\mathrm{Pre}(F)$ as the union of $\{u \in V_E \mid \exists (u, v) \in E, v \in F\}$ and $\{u \in V_A \mid \forall (u, v) \in E, v \in F\}$. The attractor sequence is the step-by-step computation of the least fixpoint of the monotone function $X \mapsto F \cup \mathrm{Pre}(X)$:

$$\begin{cases} \mathrm{Attr}^{\mathrm{E}}_0(F) = F \\ \mathrm{Attr}^{\mathrm{E}}_{k+1}(F) = \mathrm{Attr}^{\mathrm{E}}_k(F) \cup \mathrm{Pre}(\mathrm{Attr}^{\mathrm{E}}_k(F)) \end{cases}$$

The sequence $(\mathrm{Attr}^{\mathrm{E}}_k(F))_{k \geq 0}$ is increasing with respect to set inclusion, so it has a limit, denoted $\mathrm{Attr}^{\mathrm{E}}(F)$, the attractor to F. An attractor strategy to $F \subseteq V$ for Eve is a memoryless strategy that ensures from $\mathrm{Attr}^{\mathrm{E}}(F)$ to reach F within a finite number of steps. Specifically, an attractor strategy to F from $\mathrm{Attr}^{\mathrm{E}}_N(F)$ ensures to reach F within the next N steps.

ω-regular conditions. We define the Büchi and parity conditions. We equip the arena with a coloring function $c : V \to [d]$ where $[d] = \{0, \ldots, d\}$ is the set of *colors* or *priorities*. For a play π, let $\mathrm{Inf}(\pi) \subseteq [d]$ be the set of colors that appear infinitely often in π. The parity condition is defined by $\mathrm{Parity}(c) = \{\pi \mid \min(\mathrm{Inf}(\pi)) \text{ is even}\}$, *i.e* it is satisfied if the lowest color visited infinitely often is even. Here, the color set $[d]$ is interpreted as a set of priorities,

even priorities being "good" and odd priorities "bad", and lower priorities preferable to higher ones. As a special case, the class of Büchi conditions are defined using the color set $[1] = \{0, 1\}$ (*i.e* $d = 1$). Setting F as $c^{-1}(0) \subseteq V$, we define Büchi$(F) = \{\pi \mid 0 \in \mathrm{Inf}(\pi)\}$, *i.e* the Büchi condition Büchi(F) requires that infinitely many times vertices in F are reached. We usually call F the Büchi set and say that a vertex is Büchi if it belongs to F. The dual is CoBüchi(F) condition, which requires that finitely many times vertices in F are reached.

ωB-conditions. We equip the arena with k counters and an update function $C : E \to \{\varepsilon, i, r\}^k$, associating to each edge an action for each counter. The value of a counter along a play is incremented by the action i, reset by r and left unchanged by ε. We say that a counter is bounded along a play if the set of values assumed is finite, and denote by Bounded the set of plays where all counters are bounded, and Bounded(N) if bounded by N. Conditions of the form Bounded $\cap$ Parity(c) are called ωB-conditions.

Note that the bound requirement is not uniform: a strategy is winning if for all plays, there exists a bound N such that the counters are bounded by N and the parity condition is satisfied. In other words, the bound N depends on the path. The sentence "Eve wins for the bound N" means that Eve has a strategy which ensures the bound N uniformly: for all plays, the counters are bounded by the same N. Similarly, the sentence "the strategy (for Adam) breaks the bound N" means that it ensures that for all plays, either some counter reaches the value N or the parity condition is not satisfied.

Finitary conditions. Finitary conditions add bounds requirements over ω-regular conditions [3]. Given a coloring function $c : V \to [d]$, and a position k we define:

$$\mathrm{dist}_k(\pi, c) = \inf_{k' \geq k} \left\{ k' - k \mid \begin{array}{c} c(\pi_{k'}) \text{ is even, and} \\ c(\pi_{k'}) \leq c(\pi_k) \end{array} \right\};$$

i.e $\mathrm{dist}_k(\pi, c)$ is the "waiting time" by means of number of steps from the k^{th} vertex to a preferable priority (that is, even and lower). The finitary parity winning condition FinParity(c) was defined as follows in [15]: FinParity$(c) = \{\pi \mid \limsup_k \mathrm{dist}_k(\pi, c) < \infty\}$, *i.e* the finitary parity condition requires that the supremum limit of the distance sequence is bounded. In the special case where $d = 1$, this defines the finitary Büchi condition: setting $F = c^{-1}(0)$, we denote $\mathrm{dist}_k(\pi, F) = \inf\{k' - k \mid k' \geq k, \pi_{k'} \in F\}$, *i.e* $\mathrm{dist}_k(\pi, F)$ is the number of transitions followed from the k^{th} vertex before reaching the next Büchi vertex. Then FinBüchi$(F) = \{\pi \mid \limsup_k \mathrm{dist}_k(\pi, F) < \infty\}$. In the context of finitary conditions, the sentence "the strategy (for Adam) breaks the bound N" means that the strategy ensures that for all plays, there exists a position k such that $\mathrm{dist}_k(\pi, c) > N$.

▶ Remark. As defined, finitary conditions do not form a subclass of ωB-conditions; however, one can easily show that there exists a deterministic ωB-automaton which recognize FinParity(c) (see *e.g* [13]), so finitary parity games easily reduce to ωB games by composing with this deterministic automaton.

▶ **Example 1.** We conclude this section by an example witnessing the difference between playing a Büchi condition and a finitary Büchi condition over an infinite arena. This is in contrast to the case of finite arenas, where winning for Büchi and finitary Büchi conditions are equivalent. Figure 1 presents an infinite arena where only Adam has moves; he loses the Büchi game but wins for the finitary Büchi game. We give two representations: on the left as a pushdown game (defined in Section 4), and on the right as an infinite-state game.

A play consists in rounds, each starting whenever the pebble hits the leftmost vertex. In a round, Adam follows the top path, remaining in Büchi vertices; he may decide at any point to follow an edge down, following the bottom Büchi-free path before geting back to

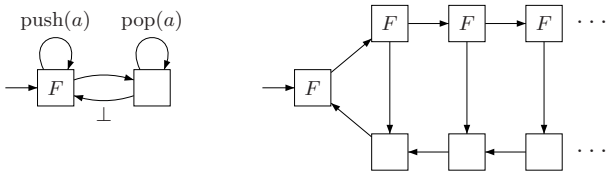

Figure 1 Adam loses the Büchi game but wins the finitary Büchi game.

the leftmost vertex. Whatever Adam does, infinitely many Büchi vertices will be visited, so Adam loses the Büchi game. However, by going further and further to the right (*e.g* for i steps in the i^{th} round), Adam ensures longer and longer paths without Büchi vertices, hence wins the finitary Büchi game.

3 Strategy complexity for finitary conditions over infinite-state games

In this section we give characterizations of the winning regions for finitary conditions over infinite arenas, and use them to establish the strategy complexity.

Our motivation to prove the existence of finite-memory strategies comes from automata theory, where several constructions rely on the existence of memoryless winning strategies (for parity games): for instance to complement tree automata [22], or to simulate alternating two-way tree automata by non-deterministic ones [36]. For this, one needs to prove the existence of finite-memory strategies whose size only depends on the condition. We present such a result in the following theorem.

▶ **Theorem 2** (Strategy complexity for finitary games). *The following assertions hold:*

1. *For all finitary Büchi games, there exists a memoryless winning strategy for Eve from her winning set.*
2. *For all finitary parity games, there exists a winning strategy for Eve from her winning set that uses at most $d/2+1$ memory states, where d is the number of colors.*

3.1 Bounded and uniform conditions

To obtain Theorem 2, we take five steps, summarized in Figure 2, which involve two variants of finitary conditions: uniform and bounded.

Figure 2 Results implications.

Uniform conditions. The bound $N \in \mathbb{N}$ is made explicit; for instance the uniform Büchi condition is $\text{Büchi}(F, N) = \{\pi \mid \limsup_k \text{dist}_k(\pi, F) \leq N\}$.

Bounded conditions. The requirement is not in the limit, but from the start of the play, *i.e* the distance function is bounded rather than eventually bounded; for instance the bounded parity condition is $\text{BndParity}(c) = \{\pi \mid \sup_k \text{dist}_k(\pi, c) < \infty\}$.

The two variants can be combined, for instance the bounded uniform Büchi condition is $\text{BndBüchi}(F, N) = \{\pi \mid \sup_k \text{dist}_k(\pi, F) \leq N\}$. Let us point out that in the special case of Büchi conditions, we have $\text{BndBüchi}(F) = \text{FinBüchi}(F)$, hence we can refer to these conditions either as bounded Büchi or as finitary Büchi.

3.2 Strategy complexity for bounded uniform Büchi games

Our first step is the study of bounded uniform Büchi games.

▶ **Theorem 3** (Strategy complexity for bounded uniform Büchi games). *For all bounded uniform Büchi games with bound N, there exists a memoryless winning strategy for Eve from her winning set.*

We show that Eve's winning set can be described using a greatest fixpoint, which allows to define a winning memoryless strategy. We define a sequence $(Z_k)_{k\geq 0}$ of subsets of V:

$$\begin{cases} Z_0 = V \\ Z_{k+1} = \mathrm{Attr}_N^{\mathrm{E}}(F \cap \mathrm{Pre}(Z_k)) \end{cases}$$

As this sequence is decreasing with respect to set inclusion, it has a limit[1], equivalently defined as the greatest fixpoint of the monotone function $X \mapsto \mathrm{Attr}_N^{\mathrm{E}}(F \cap \mathrm{Pre}(X))$: we denote it by Z.

▶ **Lemma 4.** $Z = \mathcal{W}_E(\mathrm{BndBüchi}(F, N))$

The crucial point is the left-to-right inclusion, which consists in constructing a winning strategy for Eve from Z. Roughly speaking, the strategy is obtained by nesting attractor strategies on the slices defined by the sets $(Z_k)_{k\geq 0}$, and this strategy is memoryless.

3.3 From bounded uniform Büchi games to finitary Büchi games

We sketch the second step. The bounded uniform Büchi conditions are the prefix-dependent counterpart of the uniform Büchi conditions: $\mathrm{Büchi}(F, N) = V^* \cdot \mathrm{BndBüchi}(F, N)$. However, this equality does not imply the equality between $\mathcal{W}_E(\mathrm{Büchi}(F, N))$ and the attractor of $\mathcal{W}_E(\mathrm{BndBüchi}(F, N))$. The following properties hold:

1. $\mathcal{W}_E(\mathrm{BndBüchi}(F, N)) \subseteq \mathcal{W}_E(\mathrm{Büchi}(F, N))$,
2. if $\mathcal{W}_E(\mathrm{BndBüchi}(F, N))$ is empty then $\mathcal{W}_E(\mathrm{Büchi}(F, N))$ is empty.

We sketch the proof of the second item. Assume that Eve wins nowhere for $\mathrm{BndBüchi}(F, N)$, then $\mathcal{W}_A(\mathrm{BndBüchi}(F, N))$ is the set of all vertices: from everywhere Adam can break the bound N once. Repeating such a strategy, he can break the bound N infinitely often, so Adam wins everywhere for the condition $\mathrm{Büchi}(F, N)$, which implies $\mathcal{W}_E(\mathrm{Büchi}(F, N)) = \emptyset$.

The two properties above imply that the uniform Büchi winning set is obtained by a least fixpoint iteration using the bounded uniform Büchi winning set. In particular, the memoryless determinacy transfers from bounded uniform Büchi to uniform Büchi.

This technique will be used several times in the paper (see *e.g* [27] for similar fixpoint iterations). It consists in decomposing the uniform Büchi winning set into a sequence of disjoint subarenas called "slices", and define a positional strategy for each slice. Aggregating all those strategies yields a positional winning strategy for the uniform Büchi condition. The first slice is the attractor of the bounded uniform Büchi winning set, for which Eve has a positional winning strategy for the uniform Büchi condition. We remove the first slice and proceed inductively with the remaining arena. The key observation is that a play consistent with the obtained strategy can only go down the slices, so eventually remains in one slice.

[1] This follows from our assumption that the arenas have a countable set of vertices. Here we could drop this assumption and define the sequence indexed by ordinals, which we avoided for the sake of readability.

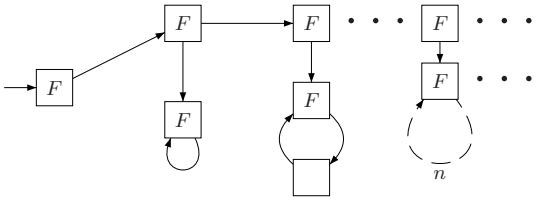

Figure 3 An infinite arena where Eve cannot predict the bound.

We sketch the third step. Denote by U the operator that associates to an arena the set of vertices $\mathrm{Attr}^{\mathrm{E}}\left(\bigcup_N \mathcal{W}_E(\text{Büchi}(F,N))\right)$. For the sake of readability, we also see U as a set of vertices once an arena is fixed. We first show how to obtain a memoryless strategy that wins for the condition $\text{FinBüchi}(F)$ from U. This requires us to compose several memoryless strategies into one:

▶ **Lemma 5** (Union and memoryless strategies [24]). *Let $(\Omega_n)_{n\in\mathbb{N}}$ be a family of Borel conditions, and assume $\cup_n\Omega_n$ is prefix-independent. If for all n, Eve wins positionally for the condition Ω_n from V_n, then she wins positionally for the condition $\cup_n\Omega_n$ from $\cup_n V_n$.*

Intuitively, U is the set of vertices where Eve has a strategy to attract in a region won for some uniform Büchi condition. That is, from some point onwards, Eve can announce a bound N and claim "I will win for the condition $\text{Büchi}(F,N)$". However, it may be that even if Eve wins, she is never able to announce a bound: such a situation happens in Example 6.

▶ **Example 6.** Figure 3 presents an infinite one-player arena, where Eve wins yet is not able to announce a bound. A loop labeled n denotes a loop of length n, where a Büchi vertex is visited every n steps. In this game, as long as Adam decides to remain in the top path, Eve cannot claim that she will win for some uniform Büchi condition.

The following properties hold:

1. $U \subseteq \mathcal{W}_E(\text{FinBüchi}(F))$,
2. if U is empty then $\mathcal{W}_E(\text{FinBüchi}(F))$ is empty.

We sketch the second item. Assume that for all N, the winning set $\mathcal{W}_E(\text{Büchi}(F,N))$ is empty, so Adam wins for the condition $\text{CoBüchi}(F,N)$ from everywhere: let τ_N be a winning strategy for Adam. From any vertex, the strategy τ_N ensures that at some point, there will be a sequence of N consecutive non-Büchi vertices. Playing in turns τ_1 until such a sequence occurs, then τ_2, and so on, ensures that the condition $\text{FinBüchi}(F)$ is spoiled. Hence Adam wins everywhere for the condition $\text{CoFinBüchi}(F)$, which implies $\mathcal{W}_E(\text{FinBüchi}(F)) = \emptyset$.

As for the second step, the winning region for finitary Büchi is obtained as the least fixpoint of the operator U, implying the memoryless determinacy for finitary Büchi.

We summarize in the following theorem the winning sets characterizations obtained for the three variants of Büchi conditions, using mu-calculus formulae with infinite disjunction.

▶ **Theorem 7** (Characterizations of the winning set).

$$\mathcal{W}_E(\text{BndBüchi}(F,N)) = \nu Z \cdot \mathrm{Attr}^{\mathrm{E}}_N(F \cap \mathrm{Pre}(Z))$$

$$\mathcal{W}_E(\text{Büchi}(F,N)) = \mu Y \cdot \nu Z \cdot \mathrm{Attr}^{\mathrm{E}}_N((F \cup Y) \cap \mathrm{Pre}(Z))$$

$$\mathcal{W}_E(\text{FinBüchi}(F)) = \mu X \cdot \left(\bigcup_{N\in\mathbb{N}} \mu Y \cdot \nu Z \cdot \mathrm{Attr}^{\mathrm{E}}_N((F \cup Y \cup X) \cap \mathrm{Pre}(Z)) \right)$$

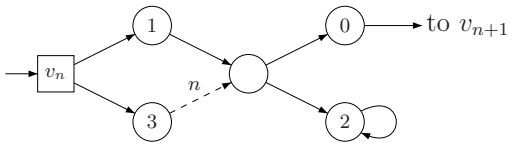

Figure 4 An infinite arena where Eve needs memory to win BndParity(c).

3.4 From finitary Büchi to finitary parity games

We sketch the fourth step, which is a reduction from bounded parity games to bounded Büchi games. We consider a coloring function $c : V \to [d]$, and assume d is even. Define the memory structure $\mathcal{M} = (\{1, 3, \ldots, d-1\} \cup \{d\}, m_0, \mu)$, where:

$$\mu(m, (v, v')) = \begin{cases} m & \text{if } c(v') \geq m \\ c(v') & \text{if } c(v') < m \text{ and } c(v') \text{ is odd} \\ d & \text{if } c(v') < m \text{ and } c(v') \text{ is even} \end{cases}$$

The initial memory state m_0 is $c(v_0)$ if $c(v_0)$ is odd, and d otherwise. Intuitively, this memory structure keeps track of the most urgent pending request.

Let $F = \{(v, d) \mid c(v) \text{ is even}\}$. We argue that $\mathcal{G} = (\mathcal{A}, \text{BndParity}(c))$ is equivalent to $\mathcal{G} \times \mathcal{M} = (\mathcal{A} \times \mathcal{M}, \text{BndBüchi}(F))$. This follows from the equivalence:

$$\pi \in \text{BndParity}(c) \quad \text{if and only if} \quad \widetilde{\pi} \in \text{BndBüchi}(F),$$

where $\widetilde{\pi}$ is the play in $\mathcal{G} \times \mathcal{M}$ corresponding to π. Since Eve has a memoryless winning strategy in any bounded Büchi game, which implies that she has a winning strategy using $\mathcal{M}$ as memory structure in the original bounded parity game $\mathcal{G}$.

▶ **Example 8.** Figure 4 presents an infinite arena, where for condition BndParity(c), Eve needs two memory states to win. This is in contrast with finite arenas, where she has memoryless winning strategies [15]. The label n on an edge indicates that the length of the path is n. A play is divided in rounds, and a round is as follows: first Adam makes a request, either 1 or 3, and then Eve either answers both requests and proceeds to the next round, or stops the play visiting color 2. Assume Eve uses a memoryless strategy, and consider two cases: either she chooses always 0, then Adam wins by choosing always 3, ensuring that the response time grows unbounded, or at some round she chooses 2, then Adam wins by choosing 1 at this particular round, ensuring that this last request will never be responded. However, if Eve answers correctly – that is choosing color 0 for the request 1, and color 2 for the request 3 – the bounded parity condition is satisfied; this requires two memory states.

The fifth step is very similar to the second, and is omitted here.

4 Pushdown ωB games

In this section we consider pushdown ωB games and prove a collapse result. Along with previous results [6, 7], this implies that determining the winner in such games is decidable.

Pushdown arenas. A pushdown process is a finite-state machine which features a stack: it is described as (Q, Γ, Δ) where Q is a finite set of control states, Γ is the stack alphabet and Δ is the transition relation. There is a special stack symbol denoted $\perp$ which does not

belong to Γ; we denote by $\Gamma_\perp$ the alphabet $\Gamma \cup \{\perp\}$. A configuration is a pair $(q, u\perp)$ (the top stack symbol is the leftmost symbol of u). There are three kinds of transitions in Δ:

- $(p, a, \text{push}(b), q)$: allowed if the top stack element is $a \in \Gamma_\perp$, the symbol $b \in \Gamma$ is pushed onto the stack.
- $(p, \text{pop}(a), q)$: allowed if the top stack element is $a \in \Gamma$, the top stack symbol a is popped from the stack.
- (p, a, skip, q): allowed if the top stack element is $a \in \Gamma_\perp$, the stack remains unchanged.

The symbol $\perp$ is never pushed onto, nor popped from the stack. The pushdown arena of a pushdown process is defined as $(Q \times \Gamma^*\perp, (Q_E \times \Gamma^*\perp, Q_A \times \Gamma^*\perp), E)$, where (Q_E, Q_A) is a partition of Q and E is given by the transition relation Δ. For instance if $(p, a, \text{push}(b), q) \in \Delta$, then $((p, aw\perp), (q, baw\perp)) \in E$, for all words w in Γ^*.

Conditions. The parity conditions for pushdown arenas are specified over the control states, *i.e* do not depend on the stack content. Formally, a coloring function is given by $c : Q \to [d]$, and extended to $c : Q \times \Gamma^*\perp \to [d]$ by $c(q, u\perp) = c(q)$.

We begin this section by giving an example witnessing an interesting phenomenon of pushdown games with ωB-conditions.

▶ **Example 9.** Figure 5 presents a pushdown ωB game where Eve wins but with arbitrarily large value of the counter, depending on Adam. Let us first look at the two bottom states: in the left-hand state at the bottom, Adam can push as many b's as he wishes, and moves the token to the state to its right, where all those b's are popped one at a time, incrementing the counter each time. In other words, each visit of the two bottom states allows Adam to announce a number N and to increment the counter by N. We now look at the states on the top line: the initial state is the leftmost one, where Adam can push an arbitrary number of a's. We see those a's as credits: from the central state, Adam can use one credit (*i.e* pop an a) to pay a visit to the two bottom states. When he runs out of credit, which will eventually happen, he moves the token to the rightmost state, where nothing happens anymore.

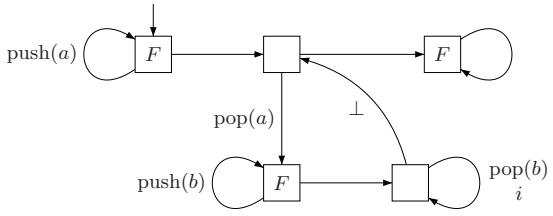

Figure 5 A pushdown ωB game where Eve wins but with arbitrarily large counter value.

$\mathcal{P}$-automata. We will use alternating $\mathcal{P}$-automata to recognize sets of configurations: an alternating $\mathcal{P}$-automaton $\mathcal{B} = (S, \delta, F)$ for the pushdown process (Q, Γ, Δ) is a classical alternating automaton over finite words: S is a finite set of control states, $\delta : S \times \Gamma \to \mathcal{B}^+(S)$ is the transition function (the notation $\mathcal{B}^+(S)$ denotes the positive boolean formulae over S) and F is the subset of S of final states. We assume that the set of states S contains Q. A configuration $(q, u\perp)$ is accepted by $\mathcal{B}$ if it is accepted with $q \in Q \subseteq S$ as initial state and the classical alternating semantics. A set of configurations is called *regular* if it is accepted by an alternating $\mathcal{P}$-automaton.

The following theorem shows that the winning region is regular for a wide class of conditions [34, 35].

▶ **Theorem 10** ([35]). *For all pushdown games, for all winning conditions $\Omega \subseteq Q^\omega$ that are Borel and prefix-independent, the set $\mathcal{W}_E(\Omega)$ is a regular set of configurations recognized by an alternating $\mathcal{P}$-automaton of size $|Q|$.*

4.1 The collapse result

We denote that LimitBounded(N) the set of plays which contain a suffix for which the counters are bounded by N.

▶ **Theorem 11** (The forgetful property). *For all pushdown ωB games, for all initial configurations, the following are equivalent:*

- *$\exists \sigma$ strategy for Eve, $\forall \pi$ plays, $\exists N \in \mathbb{N}$, $\pi \in \text{Bounded}(N) \cap \text{Parity}(c)$,*
- *$\exists \sigma$ strategy for Eve, $\exists N \in \mathbb{N}$, $\forall \pi$ plays, $\pi \in \text{LimitBounded}(N) \cap \text{Parity}(c)$.*

The intuition behind the name forgetful property is the following: even if a configuration carries an unbounded amount of information (since the stack may be arbitrarily large), this information cannot be forever transmitted along a play. Indeed, to increase the counter values significantly, Adam has to use the stack, consuming or forgetting its original information. Example 9 shows that the content of the stack can be used as "credit" for Adam, but also that if Eve wins then from some point onwards this credit vanishes.

We sketch the proof of the forgetful property. We abbreviate $\mathcal{W}_E(\text{LimitBounded}(N) \cap \text{Parity}(c))$ by $\mathcal{W}_E(N)$ and $\mathcal{W}_E(\text{Bounded} \cap \text{Parity}(c))$ by $\mathcal{W}_E$. The following properties hold:

1. $\mathcal{W}_E(0) \subseteq \mathcal{W}_E(1) \subseteq \mathcal{W}_E(2) \subseteq \cdots \subseteq \mathcal{W}_E$.
2. There exists N such that $\mathcal{W}_E(N) = \mathcal{W}_E(N+1) = \cdots$.
3. For such N, we have $V \setminus \mathcal{W}_E(N) \subseteq \mathcal{W}_A$, hence $\mathcal{W}_E = \mathcal{W}_E(N)$.

The first item is clear. For the second we rely on Theorem 10. For every N there exists $\mathcal{B}_N$ an alternating $\mathcal{P}$-automaton of size $|Q|$ recognizing $\mathcal{W}_E(N)$. Since there are finitely many alternating $\mathcal{P}$-automata of size $|Q|$, the increasing sequence of the set of configurations they recognize is ultimately constant, *i.e* there exists N such that $\mathcal{B}_N = \mathcal{B}_{N+1} = \ldots$. We now argue that the third item holds. From the complement of $\mathcal{W}_E(N)$, Adam can ensure to break the bound N, but also $N+1$, and so on, yet remaining there. Iterating such strategies ensures that the ωB-condition is spoiled, which concludes the proof.

▶ Remark. The above proof does not give a bound on N; indeed, the sequence $(\mathcal{B}_N)_{N \in \mathbb{N}}$ is ultimately constant, but the fact that two consecutive automata recognize the same set of configurations does not imply that from there on the sequence is constant. It follows that N can be *a priori* arbitrarily large.

We refer to [14] for examples showing that the bound N is at least doubly-exponential in the number of vertices, and exponential in the stack alphabet.

4.2 Decidability of pushdown ωB games

We give two proofs of decidability of solving pushdown games:

- First, we prove the decidability of solving pushdown finitary games, relying on the finite-memory results of Section 3 (Theorem 2), the collapse result (Theorem 11), and [7].
- Second, we prove the decidability of solving pushdown ωB games, generalizing the first item. This relies on the collapse result (Theorem 11) and [6].

We begin by proving the decidability of pushdown finitary games. Note that the second property in Theorem 11, namely:

$$\exists \sigma \text{ strategy for Eve}, \exists N \in \mathbb{N}, \forall \pi \text{ plays}, \ \pi \in \text{LimitBounded}(N) \cap \text{Parity}(c) \ ,$$

can be written as an existential bounding formula over infinite trees, whose satisfiability was proved decidable in [7]. This relies on an MSO interpretation of pushdown graphs into infinite trees, following [31]. Indeed, thanks to Theorem 2, Eve has a memoryless winning strategy in $\mathcal{G} \times \mathcal{M} = (\mathcal{A} \times \mathcal{M}, \text{FinParity}(c))$ from her winning set, and such a strategy can be described as a set of edges, hence as a monadic second-order variable.

The second proof relies on [6], where it is shown that the membership problem for two-way alternating parity cost-automata over regular trees is decidable; we reduce our problem to this. The first step is to reduce the problem of solving a pushdown ωB game to the membership problem for two-way alternating ωB automata over regular trees, following [28]. Let $\mathcal{A}$ be the two-way alternating ωB automaton obtained and t the regular tree. Now Theorem 11 implies that Eve wins the pushdown ωB game if and only if:

$$\exists N \in \mathbb{N}, \exists \sigma \text{ strategy for Eve}, \forall \pi \text{ plays}, \ \pi \in \text{LimitBounded}(N) \cap \text{Parity}(c) \ .$$

We construct a two-way alternating cost-automaton $\mathcal{A}'$ from $\mathcal{A}$ such that $\mathcal{A}'$ accepts t (as a cost-automaton) if and only if $\mathcal{A}$ accepts t (as an ωB automaton). The automaton $\mathcal{A}'$ is obtained by adding at each step the ability to reset all counters at the price of visiting a very bad color for the parity condition, which takes care of the difference between LimitBounded(N) and Bounded(N).

The main result of this section follows:

▶ **Theorem 12.** *Solving a pushdown ωB-game is decidable.*

5 Pushdown games with finitary and stack boundedness conditions

In this section, we consider pushdown games with finitary parity along with stack boundedness conditions, following [10, 24]. We prove that solving such games is EXPTIME-complete. This is achieved by a reduction which relies on two ideas, that we present separately; the first is a reduction from finitary parity to bounded parity, and the second a collapse result for finitary Büchi along with stack boundedness conditions. We then show how to combine them to obtain a complete reduction, with an optimal complexity.

We denote by BndSt the stack boundedness condition:

$$\text{BndSt} = \{\pi \mid \exists N, \begin{array}{c}\text{all configurations in } \pi \text{ have} \\ \text{stack height less than } N\end{array}\} \ .$$

5.1 A reduction from finitary parity to bounded parity

The reduction relies on a *restart* gadget. We consider a pushdown finitary parity game, given by the coloring function $c : Q \to [d]$, where we assume d to be odd. Between every transition we add a restart gadget, where Eve can choose either to follow the transition, or to restart: this entails that first a state with priority 0 is visited, where Adam can push on the stack a new symbol $\sharp$ an arbitrary number of times, and then go to a state with priority d, where he pops all the $\sharp$ symbols from the stack, before following the original transition. The intuition is the following: whenever Eve chooses to restart, visiting the vertex with priority 0 answers all previous requests, but this comes with the cost that Adam will be able to stay for an arbitrary long time on a state of odd priority. Therefore, Eve can restart only finitely many times. The gadget is represented in Figure 6.

▶ **Lemma 13.** *Eve wins the finitary parity game if and only if she wins the reduced bounded parity game.*

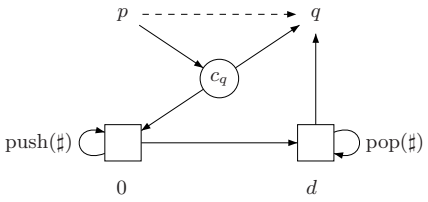

Figure 6 The restart gadget.

5.2 The special case of Büchi conditions

In the study of finitary games over finite graphs [15], the following observation is made: finitary Büchi coincide with Büchi, while finitary parity differs from parity as soon as three colors are involved. Over pushdown arenas, even finitary Büchi differs from Büchi, as noted in Example 1. Yet when intersected with the stack boundedness condition, we show that the case of finitary Büchi specializes again and collapses to Büchi.

▶ **Lemma 14.** *For all pushdown games,*

$$\mathcal{W}_E(\text{FinBüchi}(F) \cap \text{BndSt}) = \mathcal{W}_E(\text{Büchi}(F) \cap \text{BndSt}) \ .$$

The left-to-right inclusion is clear, since $\text{FinBüchi}(F) \subset \text{Büchi}(F)$. We prove the converse inclusion relying on memoryless determinacy for the condition $\text{Büchi}(F) \cap \text{BndSt}$ [10]: assume that σ is a memoryless strategy ensuring $\text{Büchi}(F) \cap \text{BndSt}$, and let π be a play consistent with σ. First note that between two visits of the same configuration, there must be a Büchi configuration, otherwise iterating this loop would be a play consistent with σ yet losing. The second observation is that since the stack height remains smaller than a bound N, the number of different configurations visited in π is finite and bounded by a function of N. The combination of these two arguments imply that π satisfies $\text{FinBüchi}(F)$.

5.3 The complete reduction

We show how to use both ideas to handle pushdown games with finitary parity and stack boundedness conditions. We present a three-step reduction, illustrated in Figure 7.

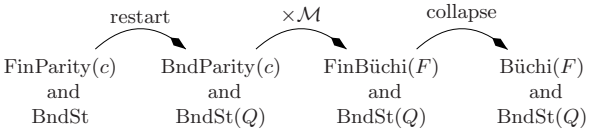

Figure 7 Sequence of reductions.

The first step is to adapt the reduction from finitary parity to bounded parity, now intersected with the stack boundedness condition. To this end, we need to modify the stack boundedness condition so that it ignores the configurations in the restart gadget; we define its restriction to Q:

$$\text{BndSt}(Q) = \{\pi \mid \exists N, \begin{array}{c}\text{all configurations in } \pi \\ \text{with control state in } Q \\ \text{have stack height less than } N\end{array}\} \ .$$

Now the reduction is from finitary parity and stack boundedness to bounded parity and restricted stack boundedness.

The second step is to compose with the memory structure from the fourth step of Section 3, giving an equivalent pushdown game with the condition finitary Büchi and restricted stack boundedness.

The third step is the collapse of finitary Büchi to Büchi. Note that the collapse stated in Lemma 14 deals with stack boundedness, not restricted to a subset of states. Indeed, the result does not hold in general for this modified stack boundedness condition, but it does hold here due to the special form of the restart gadget, that is used only finitely many times.

This three-step reduction produces in linear time an equivalent pushdown game with the condition Büchi and stack boundedness restricted to Q. It has been shown in [10, 24] that deciding the winner in a pushdown game with condition Büchi and stack boundedness is EXPTIME-complete; a slight modification of their techniques extends this to the restricted definition of stack boundedness.

▶ **Theorem 15.** *Determining the winner in a pushdown game with finitary parity and stack boundedness conditions is* EXPTIME*-complete.*

Conclusion. We studied boundedness games over infinite arenas, and investigated two questions. First, the strategy complexity over general infinite arenas; we proved that finite-memory winning strategies exist for finitary parity games. It remains open to extend this to cost-parity games [23]. Second, the decidability of pushdown games; we proved that pushdown ωB-games are decidable, and pushdown games with finitary parity along with stack boundedness conditions are EXPTIME-complete.

Acknowledgments. We thank Denis Kuperberg and Thomas Colcombet for sharing and explaining [6], Damian Niwinski for raising the question of pushdown finitary games, Olivier Serre for many inspiring discussions and Florian Horn for interesting suggestions. We are grateful to the anonymous reviewers for their valuable comments.

References

1 P. A. Abdulla, A. Bouajjani, and J. d'Orso. Monotonic and downward closed games. *J. Log. Comput.*, 18(1):153–169, 2008.
2 B. Alpern and F. B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985.
3 R. Alur and T. A. Henzinger. Finitary fairness. *ACM Trans. Program. Lang. Syst.*, 20(6):1171–1194, 1998.
4 R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
5 R. Alur, S. L. Torre, and P. Madhusudan. Modular strategies for recursive game graphs. *Theor. Comput. Sci.*, 354(2):230–249, 2006.
6 A. Blumensath, T. Colcombet, D. Kuperberg, and M. V. Boom, 2013. Personal communication.
7 M. Bojańczyk. A bounding quantifier. In *CSL*, pages 41–55, 2004.
8 M. Bojańczyk and T. Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296, 2006.
9 M. Bojańczyk and S. Toruńczyk. Weak MSO+U over infinite trees. In *STACS*, pages 648–660, 2012.
10 A.-J. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with unboundedness and regular conditions. In *FSTTCS*, pages 88–99, 2003.

11 T. Brázdil, P. Jancar, and A. Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.

12 J. R. Büchi and L. H. Landweber. Definability in the monadic second-order theory of successor. *J. Symb. Log.*, 34(2):166–170, 1969.

13 K. Chatterjee and N. Fijalkow. Finitary languages. In *LATA*, pages 216–226, 2011.

14 K. Chatterjee and N. Fijalkow. Infinite-state games with finitary conditions. *CoRR*, abs/1301.2661, 2013.

15 K. Chatterjee, T. A. Henzinger, and F. Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009.

16 T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, pages 139–150, 2009.

17 T. Colcombet. Fonctions régulières de coût. Habilitation Thesis (in French), 2013.

18 T. Colcombet, D. Kuperberg, and S. Lombardy. Regular temporal cost functions. In *ICALP (2)*, pages 563–574, 2010.

19 T. Colcombet and C. Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.

20 T. Colcombet and C. Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79, 2010.

21 E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *FOCS*, pages 328–337, 1988.

22 E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377, 1991.

23 N. Fijalkow and M. Zimmermann. Cost-parity and cost-streett games. In *FSTTCS*, pages 124–135, 2012.

24 H. Gimbert. Parity and exploration games on infinite graphs. In *CSL*, pages 56–70, 2004.

25 E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

26 Y. Gurevich and L. Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982.

27 E. Kopczyński. Half-positional determinacy of infinite games. In *ICALP (2)*, pages 336–347, 2006.

28 O. Kupferman and M. Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *CAV*, pages 36–52, 2000.

29 D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

30 R. McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993.

31 D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.

32 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

33 M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969.

34 O. Serre. Note on winning positions on pushdown games with ω-regular conditions. *Inf. Process. Lett.*, 85(6):285–291, 2003.

35 O. Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris 7 - Denis Diderot, 2006.

36 M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.

37 I. Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.