Infinite-state games with finitary conditions

Krishnendu Chatterjee, Nathanaël Fijalkow

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, CNRS & Université Denis Diderot - Paris 7, France
Institute of Informatics, University of Warsaw, Poland
nath@liafa.univ-paris-diderot.fr

Abstract We study two-player zero-sum games over infinite-state graphs with boundedness 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 games, and finite-memory suffices for finitary parity games.

We then study pushdown boundedness games, with two contributions. First we prove a collapse result for pushdown ωB games, 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.

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 [20,21,24,25,38]. 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 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, strengthens 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.

Bounds in ω-regularity. The finitary conditions are closely related to the line of work initiated by Bojańczyk in [7], where the $\mathrm{MSO} + \mathbb{B}$ logic was defined, generalizing MSO 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 $\mathrm{MSO} + \mathbb{B}$ 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 was shown that automata with finitary conditions exactly correspond to star-free ωB-expressions. Moreover, the finitary conditions are recognized by ωB-automata, hence they can be considered as a subcase of ωB-conditions.

Regular cost-functions. A different perspective for bounds in ω-regularity was developed by Colcombet in [15] 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 [19], but its extension over infinite trees is still open, and would imply the decidability of the index of the non-deterministic Mostowski hierarchy [18], a problem open for decades. A subclass of cost-MSO called temporal cost logic was introduced in [17] and is the counterpart of finitary conditions for regular cost-functions [13], also reminiscent of desert automata [26].

Quantification order. The essential difference between the approaches underlying the logics $\mathrm{MSO} + \mathbb{B}$ and cost-MSO is a quantifier switch. We illustrate this in the context of games: a typical property expressed in $\mathrm{MSO} + \mathbb{B}$ 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, $\mathrm{MSO} + \mathbb{B}$ expresses non-uniform bounds while bounds in cost-MSO are uniform.

Solving boundedness games. Games over finite graphs with finitary conditions have been studied in [14], leading to very efficient algorithms: finitary parity games can be solved in polynomial time (unlike classical parity games). 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 [10,23]. 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.

Memoryless determinacy for infinite-state games. Our motivation to prove the existence of finite-memory winning strategies is towards automata theory, where several constructions rely on the existence of memoryless winning strategies (for parity games): for instance to complement tree automata [21], or to simulate alternating two-way tree automata by non-deterministic ones [36].

In particular, Colcombet pointed out in [16] that the remaining difficulty to establish the decidability of cost-MSO over infinite trees is a good understanding of boundedness games, and more specifically the cornerstone is to extend the memoryless determinacy of parity games over infinite graphs, following [20,21,25].

Our contributions. We study two questions about infinite-state games with boundedness 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 games memoryless strategies suffice, and that for finitary parity games, memory of size $\ell + 1$ suffices, where ℓ is the number of odd priorities in the parity condition.

Pushdown games. We present two contributions.

First we consider pushdown boundedness games and prove that the following statements are equivalent: "there exists a strategy, such that for all plays, there exists a bound on the counter values and the parity condition is satisfied" and "there exists a strategy, there exists a bound N, such that for all plays, *eventually* the counter values are bounded by N and the parity condition is satisfied". 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 solving 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.

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 interaction results a path in the graph, called a *play* and usually denoted $\pi = v_0, v_1, \ldots$. To avoid the nuisance of dealing with finite plays, we assume that the graphs have no dead-ends: all vertices have an outgoing edge, so the plays are infinite. 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. Given an arena $\mathcal{A}$, a subset U of vertices induces a subarena if all vertices in U have an outgoing edge to U. We denote by $\mathcal{A}[U]$ the induced arena.

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 has a winning strategy from U" means that she has a strategy such that for all initial vertex v_0 in U, 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 where Eve wins, also referred as winning set, or winning region, and analogously $\mathcal{W}_A(\mathcal{G})$ for Adam. Whenever the arena $\mathcal{A}$ is clear from the context, we use $\mathcal{W}_E(\Omega)$ instead of $\mathcal{W}_E(\mathcal{A}, \Omega)$. A very important theorem in game theory, due to Martin [29], states that Borel games (that is, where the condition is Borel, a topological condition) are determined, *i.e* we have $\mathcal{W}_E(\mathcal{G}) \cup \mathcal{W}_A(\mathcal{G}) = V$: 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 $\mathcal{A}$ and an initial vertex v_0 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 $\sigma : V_E \to V$.

We can make the synchronized product explicit: an arena $\mathcal{A}$ and a memory structure $\mathcal{M}$ for $\mathcal{A}$ induce the expanded arena $\mathcal{A} \times \mathcal{M} = (V \times M, (V_E \times M, V_A \times M), E \times \mu)$ where $E \times \mu$ is defined by $((v, m), (v', m')) \in E \times \mu$ if $(v, v') \in E$ and $\mu(m, (v, v')) = m'$. There is a natural one-to-one mapping between plays in $\mathcal{A}$ and in $\mathcal{A} \times \mathcal{M}$, and also from memoryless strategies in $\mathcal{A} \times \mathcal{M}$ to strategies in $\mathcal{A}$ using $\mathcal{M}$ as memory structure. It follows that if a player has a memoryless strategy for the arena $\mathcal{A} \times \mathcal{M}$, then he has a strategy using $\mathcal{M}$ as memory structure for the arena $\mathcal{A}$, producing the same plays. This *key* property will be used throughout the paper.

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 $X \mapsto F \cup \mathrm{Pre}(X)$:

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

The sequence $(\mathrm{Attr}_k^{\mathrm{E}}(F))_{k \geq 0}$ is increasing with respect to set inclusion, so it has a limit[1], 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 $\mathrm{Attr}_N^{\mathrm{E}}(F)$ 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. The parity conditions are self-dual, meaning that the completement of a parity condition is another parity condition: $\mathrm{CoParity}(c) = \Pi \setminus \mathrm{Parity}(c) = \mathrm{Parity}(c + 1)$. As a special case, the class of Büchi conditions are defined using the color set $[1] = \{0, 1\}$ (*i.e* $d = 1$). We define the Büchi set F as $c^{-1}(0) \subseteq V$, say that a vertex is Büchi if it belongs to F, and define $\text{Büchi}(F) = \{\pi \mid 0 \in \mathrm{Inf}(\pi)\}$, *i.e* the Büchi condition $\text{Büchi}(F)$ requires that infinitely many times vertices in F are reached.

The dual is $\text{CoBüchi}(F)$ 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 every 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 $\mathrm{Bounded}(N)$ if bounded by N. The conditions of the form $\mathrm{Bounded} \cap \mathrm{Parity}(c)$ are called ωB-conditions.

Note that the bound requirement for ωB-conditions 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) fools 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:

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

[1] Here we use the assumption that the set of vertices is countable. We could drop this assumption and define the sequence indexed by ordinals, which we avoided for the sake of readability.

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 $\mathrm{FinParity}(c)$ was defined as follows in [14]: $\mathrm{FinParity}(c) = \{\pi \mid \limsup_k \mathrm{dist}_k(\pi, c) < \infty\}$, *i.e* the finitary parity condition requires that the supremum limit of the distance sequence is bounded. A good intuition is to see the finitary parity condition as bounding the waiting time between requests, which are odd priorities, and responses, which are even priorities. In this terminology, the priority 3 is a request, answered by 0 and 2 since they are smaller, but not by 4. The finitary parity condition is satisfied by a play if there exists $N \in \mathbb{N}$ such that from some point onwards, all requests are answered within N steps.

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 steps from the k^{th} vertex to the next Büchi vertex. (Note that this is consistent with the previous notation $\mathrm{dist}_k(\pi, c)$.) Then $\mathrm{FinBüchi}(F) = \{\pi \mid \limsup_k \mathrm{dist}_k(\pi, F) < \infty\}$. In the context of finitary conditions, the sentence "the strategy (for Adam) fools the bound N" means that the strategy ensures that for all plays, there exists a position k such that $\mathrm{dist}_k(\pi, c) > N$.

We shall refer to games with ωB-conditions as ωB games, and the same applies for all kinds of conditions.

Remark 1. As defined, finitary conditions do not form a subclass of ωB-conditions; however, there exists a deterministic ωB-automaton which recognizes $\mathrm{FinParity}(c)$, so finitary games reduce to ωB games by composing with this deterministic automaton. We informally describe this automaton: it has a counter for each odd priority, and keeps track of the set of open requests. As long as a request is open, the corresponding counter is incremented at each step, and it is reset whenever the request is answered.

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 graph. This is in contrast to the case of finite graphs, where winning for Büchi and finitary Büchi conditions are equivalent. Figure 1 presents an infinite graph 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 graph (defined in Section 4), and on the right explicitely as an infinite-state graph.

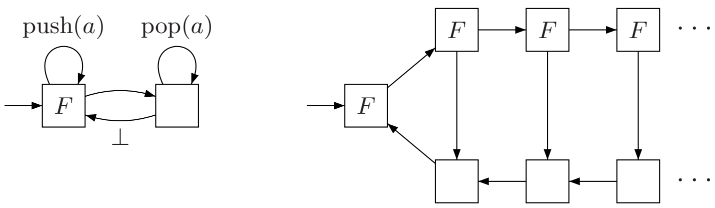

Figure 1. Adam loses the Büchi game but wins the finitary Büchi game.

A play consists in rounds, each starting whenever the pebble hits the leftmost vertex. In a round, Adam chooses a number N and follows the top path for N steps, remaining in Büchi vertices; then he goes down, and follows a path of length N without Büchi vertices, before getting back to the leftmost vertex. Whatever Adam does, infinitely many Büchi vertices will be visited, so Adam loses the Büchi game. However, by describing an unbounded sequence (*e.g* for N steps in the N^{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 for both players. The main results are summarized in the following theorem.

Theorem 1 (Strategy complexity for finitary games). *The following assertions hold:*

1. *For all finitary Büchi games, Eve has a memoryless winning strategy from her winning set.*
2. *For all finitary parity games, Eve has a finite-memory winning strategy from her winning set that uses at most $\ell + 1$ memory states, where ℓ is the number of odd colors.*

3.1 Bounded and uniform conditions

To obtain Theorem 1, we take five steps, summarized in Figure 2, which involve two variants of finitary conditions: uniform and bounded.

bounded uniform Büchi → uniform Büchi → finitary Büchi → bounded parity → finitary parity

Figure 2. Results implications.

Uniform conditions. Unlike finitary 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. Unlike finitary 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 defined as $\text{BndBüchi}(F, N) = \{\pi \mid \sup_k \text{dist}_k(\pi, F) \leq N\}$. 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 Constructing positional strategies

We start with two general techniques to construct positional strategies. Both techniques are about composing several positional strategies into one. The first lemma deals with union.

Lemma 1 (Union and positional strategies [23]). *Let $\mathcal{A}$ be an arena and $(\Omega_n)_{n \in \mathbb{N}}$ be a family of Borel conditions. If $\cup_{n \in \mathbb{N}} \Omega_n$ is prefix-independent and for all $n \in \mathbb{N}$, Eve has a positional winning strategy for the condition Ω_n from V_n, then she has a positional winning strategy for the condition $\cup_{n \in \mathbb{N}} \Omega_n$ from $\cup_{n \in \mathbb{N}} V_n$.*

Proof. We denote by Ω the condition $\cup_{n \in \mathbb{N}} \Omega_n$.

For all $n \in \mathbb{N}$, let σ_n be a positional strategy winning from V_n for the condition Ω_n. We construct σ positional strategy on $\cup_{n \in \mathbb{N}} V_n$: for $v \in \cup_{n \in \mathbb{N}} V_n$, we define $\sigma(v) = \sigma_k(v)$ where k is the smallest integer such that $v \in V_k$. Consider a play π consistent with σ from $\cup_{n \in \mathbb{N}} V_n$: it can be decomposed into finitely many infixes, each consistent with some strategy σ_k. Furthermore, the index k decreases along the play, hence is ultimately constant, so π is ultimately consistent with some σ_k. Since Ω is prefix-independent, π is in Ω, hence σ is a positional winning strategy from $\cup_{n \in \mathbb{N}} V_n$ for the condition Ω. ∎

The second lemma is about fixpoint iteration.

Lemma 2 (Fixpoint and positional strategies). *Let $\mathcal{G} = (\mathcal{A}, \Omega)$ be a game, where Ω is Borel and prefix-independent. If there exists an operator Ξ which associates to each subarena $\mathcal{A}'$ of $\mathcal{A}$ a subset of vertices of $\mathcal{A}'$ satisfying the following properties, for all subarenas $\mathcal{A}'$:*

1. *$\Xi(\mathcal{A}') \subseteq \mathcal{W}_E(\mathcal{A}', \Omega)$.*

2. If $\mathcal{W}_E(\Omega)$ is non-empty then $\Xi(\mathcal{A}')$ is non-empty.
3. Eve has a positional winning strategy from $\Xi(\mathcal{A}')$ in the game $(\mathcal{A}', \Omega)$.

then Eve has a positional winning strategy for Eve from her winning set in $\mathcal{G}$.

This technique will be used several times in the paper (see *e.g* [27] for similar fixpoint iterations). It consists in decomposing the winning set for Ω 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 Ω.

Proof. We define by induction the following objects:

- a sequence $(\mathcal{A}_k)_{k\geq 0}$ of subarenas of $\mathcal{A}$,
- a sequence of slices $(S_k)_{k\geq 1}$,
- a sequence of positional strategies $(\sigma_k)_{k\geq 1}$ for Eve from $\mathrm{Attr}^{\mathrm{E}}(S_k)$.

The first arena $\mathcal{A}_0$ is $\mathcal{A}$. Having defined $\mathcal{A}_k$, we set $S_{k+1} = \mathrm{Attr}^{\mathrm{E}}(\Xi(\mathcal{A}_k))$, $\mathcal{A}_{k+1} = \mathcal{A}_k \setminus S_{k+1}$ and σ_{k+1} as an attractor strategy on $\mathrm{Attr}^{\mathrm{E}}(\Xi(\mathcal{A}_k)) \setminus \Xi(\mathcal{A}_k)$ and a positional winning strategy from $\Xi(\mathcal{A}_k)$ in the game $(\mathcal{A}_k, \Omega)$.

First observe that the union S of all slices is the winning region for Eve in $\mathcal{G}$, this follows from *1.* and *2.*. Denote by σ the union of all strategies σ_k (note that the slices are pairwise disjoint). The second key observation is that a play consistent with σ from S can only go down the slices, so eventually remains in one slice, hence is eventually consistent with some σ_k, and as a consequence is in Ω. Thus σ is a positional winning strategy from Eve's winning region in $\mathcal{G}$. ∎

3.3 Strategy complexity for bounded uniform Büchi games

Our first step is the study of bounded uniform Büchi games. In this subsection, we obtain the following results:

Proposition 1 (Strategy complexity for bounded uniform Büchi games). *For all bounded uniform Büchi games with bound N, the following assertions hold:*

1. *Eve has a positional winning strategy from her winning set.*
2. *Adam has a finite-memory winning strategy with N memory states from his winning set.*
3. *In general, winning strategies for Adam require at least $N-1$ memory states, even over finite arenas, for $N \geq 3$.*

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

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

This sequence is decreasing with respect to set inclusion, so it has a limit denoted by Z, equivalently defined as the greatest fixpoint of the monotone function $X \mapsto \mathrm{Attr}^{\mathrm{E}}_N(F \cap \mathrm{Pre}(X))$.

Lemma 3.

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

Proof. We prove both inclusions.

- We first show that $Z \subseteq \mathcal{W}_E(\mathrm{BndBüchi}(F, N))$. Let σ^N be a positional strategy that ensures from $\mathrm{Attr}^{\mathrm{E}}_N(F \cap \mathrm{Pre}(Z))$ to reach $F \cap \mathrm{Pre}(Z)$ within N steps. We define a strategy σ on Z by:

$$\sigma(v) = \begin{cases} \sigma^N(v) & \text{if } v \in \mathrm{Attr}^{\mathrm{E}}_N(F \cap \mathrm{Pre}(Z)) \setminus F \cap \mathrm{Pre}(Z) \\ v' \in Z & \text{if } v \in F \cap \mathrm{Pre}(Z) \end{cases}$$

Consider $\pi = v_0 v_1 \dots$ a play starting from $v_0 \in Z$ consistent with σ. By definition of σ^N it will reach $F \cap \mathrm{Pre}(Z)$ within N steps, say at vertex v_{k_0} for $0 \leq k_0 \leq N$. Furthermore the play $v_{k_0+1} \dots$ is consistent with σ and starts from $v_{k_0+1} \in Z$, so repeating this reasoning by induction, we show that π visits F infinitely often, and that the distance to the next Büchi vertex remains smaller than N. Thus σ is a positional winning strategy for $\mathrm{BndBüchi}(F, N)$ from Z.

– We now show that $V \setminus Z \subseteq \mathcal{W}_A(\mathrm{BndBüchi}(F, N))$. Consider a vertex v not in Z, we define its rank to be the smallest k such that v does not belong to Z_k; note that the rank cannot be 0. A vertex of rang $k+1$ belongs to Z_k but not to Z_{k+1}. For each k we define a strategy τ_k:
  - For $k \neq 0$, the strategy τ_k ensures that from $V \setminus \mathrm{Attr}^{\mathrm{E}}_N(F \cap \mathrm{Pre}(Z_k))$, if a Büchi vertex v is reached within N steps, then it does not belong to $\mathrm{Pre}(Z_k)$.
  - For $k = 0$, the strategy τ_0 ensures that from $V \setminus \mathrm{Attr}^{\mathrm{E}}_N(F)$, no Büchi vertex is reached within N steps.

  We now define a strategy τ from Z: from a vertex of rank $k+1$, play consistently with τ_k for N steps or until a Büchi vertex v is reached, whichever comes first. In the first case, Adam wins, and in the second, by definition of τ_k, v is not in $\mathrm{Pre}(Z_k)$. Either v belongs to Eve and any successor will be in $V \setminus Z_k$, or it belongs to Adam and the strategy τ chooses a successor in $V \setminus Z_k$. Denote v' the successor of v: since it is in $V \setminus Z_k$, it has a smaller rank than v. From this vertex v', restart from scratch.

  We argue that τ is a winning strategy from $V \setminus Z$. Indeed, consider a play $\pi = v_0 v_1 \dots$ from $V \setminus Z$ consistent with τ. If v_0 has rank $k+1$, then either within N steps no Büchi vertices are visited (hence Adam wins) or its successor has a lower rank, and the play starting from this successor is consistent with τ. Since there is no infinite decreasing sequence of integers, the first situation occurs and the play π does not satisfy the bounded uniform Büchi condition. Hence τ is a winning strategy from $V \setminus Z$.

∎

So far, we proved that in bounded uniform Büchi games, Eve has a positional winning strategy from her winning set.

It is not clear from this characterization how to implement a winning finite-memory strategy for Adam. To prove the finite-memory determinacy for Adam, we rely on a reduction to safety games, that we present now. Define the memory structure $\mathcal{M} = (\{0, \dots, N\}, 0, \mu)$ as:

$$\mu(i, (v, v')) = \begin{cases} 0 & \text{if } v \in F \text{ or } v' \in F \\ i+1 & \text{if } i < N \text{ and } v, v' \notin F \\ N & \text{otherwise} \end{cases}$$

Intuitively, the memory structure counts the number of steps since the last visit to a Büchi vertex. Then $(\mathcal{G}, \mathrm{BndBüchi}(F, N))$ is equivalent to $(\mathcal{G} \times \mathcal{M}, \mathrm{Safety}(V \times \{0, \dots, N-1\}))$. Since in a safety game Adam has a positional winning strategy from his winning set, we deduce a finite-state winning strategy using $\mathcal{M}$ as memory structure from his winning set in $\mathcal{G}$. Moreover, a winning strategy using $\mathcal{M}$ does not make use of the additional memory state N, hence it actually uses N memory states, and not $N+1$.

Note that the positional result for Eve cannot be obtained from this reduction. The following example shows that the upper bound given above is (almost) tight.

Example 2. Figure 3 presents an arena where Adam wins for the condition $\mathrm{CoBndBüchi}(F, N+1)$ using N memory states and loses with less. Here $N \geq 2$. A play consists in repeating infinitely many times the following interaction: first, from c Adam chooses an i from $\{1, \dots, N\}$, then from $v_{\neq i}$ Eve chooses a j different from i, and follows a path of length N where only the j^{th} vertex is Büchi. Adam wins using N memory states by playing the last choice of Eve: this way, either Eve chooses a j larger than i so no Büchi vertices will be visited within $N+2$ steps, or she chooses a j smaller than i. The first case occurs infinitely many times, so the uniform Büchi condition is violated. If Adam uses less than N memory states, then there exists an i that he will never choose: Eve wins $\mathrm{BndBüchi}(F, N+1)$ by choosing i every time.

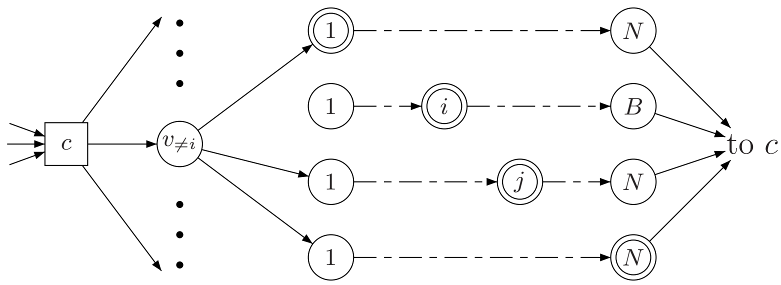

Figure 3. An arena where Adam needs N memory states to win a bounded uniform Büchi game.

3.4 Strategy complexity for uniform Büchi games

Our second step is about uniform Büchi games. In this subsection, we obtain the following results:

Proposition 2 (Strategy complexity for uniform Büchi games). *For all uniform Büchi games with bound N, the following assertions hold:*

1. *Eve has a positional winning strategy from her winning set.*
2. *Adam has a finite-memory winning strategy with $N+1$ memory states from his winning set.*
3. *In general, winning strategies for Adam require at least $N-1$ memory states, even over finite arenas, for $N \geq 2$.*

The bounded uniform Büchi conditions are the prefix-dependent counterpart of the uniform Büchi conditions:

$$\text{Büchi}(F, N) = V^* \cdot \text{BndBüchi}(F, N) \ .$$

However, this does not imply the equality between $\mathcal{W}_E(\text{Büchi}(F, N))$ and $\text{Attr}^{\text{E}}(\mathcal{W}_E(\text{BndBüchi}(F, N)))$. One inclusion holds:

$$\text{Attr}^{\text{E}}(\mathcal{W}_E(\text{BndBüchi}(F, N))) \subseteq \mathcal{W}_E(\text{Büchi}(F, N)) \ ,$$

but the other fails, as shown in Figure 4.

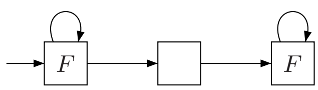

Figure 4. $\text{Attr}^{\text{E}}(\mathcal{W}_E(\text{BndBüchi}(F, 0))) \subsetneq \mathcal{W}_E(\text{Büchi}(F, 0))$

This shows that one iteration of the bounded uniform Büchi winning set does not give the whole uniform Büchi winning set. However, the following properties hold:

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

The first item is clear, we prove the second. Assume $\mathcal{W}_E(\text{BndBüchi}(F, N)) = \emptyset$, then $\mathcal{W}_A(\text{BndBüchi}(F, N)) = V$: from everywhere Adam can fool the bound N. Iterating such strategies, he can fool the bound N infinitely often, so $\mathcal{W}_A(\text{Büchi}(F, N)) = V$, which implies $\mathcal{W}_E(\text{Büchi}(F, N)) = \emptyset$.
We apply Lemma 2 with the operator Ξ that associates to each subarena $\mathcal{A}'$ the set $\mathcal{W}_E(\mathcal{A}', \text{BndBüchi}(F, N))$. The first two properties *1.* and *2.* have been proved above, and the third one is a consequence of Proposition 1,

since $V^* \cdot \mathrm{BndBüchi}(F, N) \subseteq \mathrm{Büchi}(F, N)$. It follows that in uniform Büchi games, Eve has a positional winning strategy from her winning set.

The proof of the results for Adam follows the same lines as above. We first lift up the reduction, which is now from uniform Büchi games to CoBüchi games. The memory structure is the same as above, and now $(\mathcal{G}, \mathrm{Büchi}(F, N))$ is equivalent to $(\mathcal{G} \times \mathcal{M}, \mathrm{CoBüchi}(V \times \{0, \ldots, N-1\}))$. Since in a CoBüchi game, Adam has a positional winning strategy from his winning set, we deduce a finite-state winning strategy using $\mathcal{M}$ as memory structure from his winning set in $\mathcal{G}$. Notice that this gives an upper bound of $N+1$ memory states, whereas in the case of bounded uniform Büchi games, we had an upper bound of N memory states.

We now discuss the lower bound: we can easily see that the statements about the game presented in Example 2 hold true for bounded uniform Büchi conditions as well as for uniform Büchi conditions, hence the same lower bound of $N-1$ applies.

3.5 Strategy complexity for finitary Büchi games

Our third step is about finitary Büchi games. In this subsection, we obtain the following results:

Proposition 3 (Strategy complexity for finitary Büchi games). *For all finitary Büchi games, the following assertions hold:*

1. *Eve has a positional winning strategy from her winning set.*
2. *In general winning strategies for Adam require infinite memory, even for pushdown arenas.*

Let $\mathcal{G} = (\mathcal{A}, \mathrm{FinBüchi}(F))$ be a finitary Büchi game. We denote by Ξ the operator that associates to a subarena $\mathcal{A}'$ the set of vertices $\bigcup_N \mathcal{W}_E(\mathcal{A}', \mathrm{Büchi}(F, N))$. To apply Lemma 2, we prove the following properties, for all subarenas $\mathcal{A}'$:

1. $\Xi[\mathcal{A}'] \subseteq \mathcal{W}_E(\mathcal{A}', \mathrm{FinBüchi}(F))$.
2. If $\mathcal{W}_E(\mathrm{FinBüchi}(F))$ is non-empty then $\Xi[\mathcal{A}']$ is non-empty.
3. Eve has a positional winning strategy from $\Xi[\mathcal{A}']$ in $(\mathcal{A}', \mathrm{FinBüchi}(F))$.

The first item is clear, with the following interpretation in mind: $\Xi[\mathcal{A}']$ is the set of vertices where Eve can announce a bound N upfront and claim "I will win for the condition $\mathrm{Büchi}(F, N)$". However, it may be that even if Eve wins, she is never able to announce a bound: such a situation happens in Example 3.

Example 3. Figure 5 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.

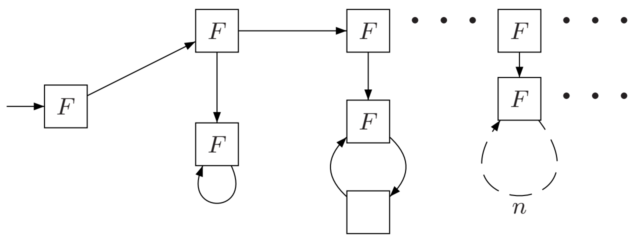

Figure 5. An infinite arena where Eve cannot predict the bound.

We prove the second item, by the contrapositive. 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 fools the bound N, *i.e* for all plays consistent with τ_N, there is 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 to spoil the condition $\text{FinBüchi}(F)$. Hence Adam wins everywhere for the condition $\text{CoFinBüchi}(F)$, which implies $\mathcal{W}_E(\text{FinBüchi}(F)) = \emptyset$.

We now prove the third item. We know from Proposition 2 that Eve has a positional winning strategy from $\mathcal{W}_E(\text{Büchi}(F, N))$ for the condition $\text{Büchi}(F, N)$. Now thanks to Lemma 1 we deduce that she has a positional winning strategy from $\bigcup_N \mathcal{W}_E(\text{Büchi}(F, N))$ for the condition $\bigcup_N \text{Büchi}(F, N)$ (that is, $\text{FinBüchi}(F)$).

By Lemma 2, in finitary Büchi games, Eve has a positional winning strategy from her winning set, and the winning region for finitary Büchi is obtained as the least fixpoint of the operator Ξ.

An arena where Adam needs infinite memory to win in a finitary Büchi game was already presented and discussed in Figure 1.

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 2 (Characterizations of the winning sets).

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

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

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

3.6 Strategy complexity for bounded parity games

Our fourth step is about bounded parity games. In this subsection, we obtain the following results:

Proposition 4 (Strategy complexity for bounded parity games). *For all bounded parity games, the following assertions hold:*

1. *Eve has a finite-memory winning strategy that uses $\ell + 1$ memory states from her winning set, where ℓ is the number of odd colors.*
2. *In general, winning strategies for Eve from her winning set require two memory states (i.e, positional strategies do not suffice for winning).*

We present a reduction from bounded parity games to bounded Büchi games. Let $\mathcal{G} = (\mathcal{A}, \text{BndParity}(c))$ be a bounded parity game equipped with the coloring function $c : V \to [d]$, and assume that d is even. Define the memory structure $\mathcal{M} = (\{1, 3, \ldots, d-1\} \cup \{d\}, m_0, \mu)$, where:

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

$$m_0 = \begin{cases} c(v_0) & \text{if } c(v_0) \text{ is odd} \\ d & \text{otherwise} \end{cases}$$

Intuitively, this memory structure keeps track of the most important pending request. It will be used several times in the paper, in Section 4 as well as in Section 5.

Let $F = \{(v,d) \mid c(v) \text{ is even}\}$. which intuitively corresponds to the case where all requests got answered. We argue that $\mathcal{G}$ is equivalent to $\mathcal{G} \times \mathcal{M} = (\mathcal{A} \times \mathcal{M}, \text{BndBüchi}(F))$, *i.e* the following are equivalent:

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

where $\widetilde{\pi}$ is the play in $\mathcal{G} \times \mathcal{M}$ corresponding to π.

We prove the left-to-right direction. Let $\pi \in \text{BndParity}(c)$, then there exists N such that for all k, $\text{dist}_k(\pi, c) \leq N$; in other words every request is answered within N steps. We argue that in $\widetilde{\pi}$, for all positions k, we have $\text{dist}_k(\widetilde{\pi}, F) \leq N \cdot \ell$. Indeed, consider the memory states assumed along $\widetilde{\pi}$, *i.e* the set of open requests. Since each request is answered within N steps, they are removed from the memory state; however, it may be that along the way other requests are opened. If they are not answered within these N steps, then they are smaller. The set of open requests can only decrease ℓ times, implying our claim.

Conversely, let $\widetilde{\pi} \in \text{BndBüchi}(F)$, then there exists N such that for all k, $\text{dist}_k(\widetilde{\pi}, F) \leq N$. in other words after N steps no request is pending. A fortiori, every request is answered within N steps, so $\pi \in \text{BndParity}(c)$. This concludes.

Thanks to Proposition 3, in a bounded Büchi game Eve has a positional winning strategy from her winning set, which implies that she has a positional winning strategy using $\mathcal{M}$ as memory structure from her winning set in $\mathcal{G}$.

Note that this does not give a reduction from finitary parity games to finitary Büchi games: the above equivalence does not hold for the prefix-independent conditions. For instance, $\pi = 1 \cdot 2^\omega$ satisfies the finitary parity condition but $\widetilde{\pi} = (1,1) \cdot (2,1)^\omega$ does not satisfy the finitary Büchi condition (the memory state remains equal to 1 forever).

We now consider the lower bounds on memory. The fact the Eve needs memory is illustrated in Example 4. Note that from the special case of bounded Büchi conditions we already know an infinite lower bound for Adam.

Example 4. Figure 6 presents an infinite arena, where for condition $\text{BndParity}(c)$, Eve needs two memory states to win. This is in contrast with finite arenas, where she has positional winning strategies [14]. 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 positional 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, and this requires two memory states.

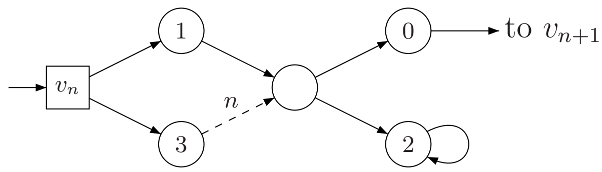

Figure 6. An infinite arena where Eve needs memory to win $\text{BndParity}(c)$.

Before proceeding to the fifth and last step, let us discuss why the fourth step was about bounded parity conditions rather than uniform ones. In both uniform parity games and bounded parity games, Eve needs memory to win; this is shown in Example 4 for bounded parity conditions, and in Example 5 for uniform parity conditions. It follows that using any of the two routes would not give positional winning strategies for our final goal, finitary parity conditions. Furthermore, extending the techniques for bounded Büchi games to bounded parity games is quite technical, as characterizing the winning regions requires nesting least and greatest fixpoints, whereas the reduction we described from bounded parity games to bounded Büchi games is both conceptually simple and effective.

Example 5. Figure 7 presents a finite arena, where for condition $\mathrm{Parity}(p, 2)$, Eve needs two memory states to win. First Adam makes a request, either 1 or 3, and then Eve chooses between 0 and 2. If Eve answers correctly – that is choosing color 0 for the request 3, and color 2 for the request 1– the bound requirement is satisfied, and this requires two memory states. Otherwise, either the bound requirement is too large (if she chooses color 0 while Adam chose color 3) or the answer is not appropriate (if she chooses color 2 while Adam chose color 1). This example is easily generalized to the case of $2d + 1$ colors, and there Eve needs $d + 1$ memory states to answer the requests appropriately.

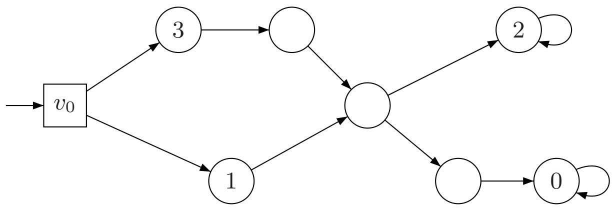

Figure 7. An arena where Eve needs memory to win $\mathrm{Parity}(p, 2)$.

3.7 Strategy complexity for finitary parity games

Our last step is about finitary parity games. In this subsection, we obtain the following results:

Proposition 5 (Strategy complexity for finitary parity games). *For all finitary parity games, Eve has a finite-memory winning strategy from her winning set that uses at most $\ell + 1$ memory states, where ℓ is the number of odd colors.*

Once again, we rely on Lemma 2 to prove this result. Let $\mathcal{G} = (\mathcal{A}, \mathrm{FinBüchi}(F))$ be a finitary parity game, and $\mathcal{M}$ the memory structure defined in the fourth step. We consider the arena $\mathcal{A} \times \mathcal{M}$, and denote by Ξ the operator that associates to a subarena $\mathcal{A}'$ of $\mathcal{A} \times \mathcal{M}$ the set of vertices $\mathcal{W}_E(\mathcal{A}', \mathrm{BndParity}(c))$. Specifically, we have, for all subarenas $\mathcal{A}'$:

1. $\Xi[\mathcal{A}'] \subseteq \mathcal{W}_E(\mathcal{A}', \mathrm{FinParity}(c))$.
2. If $\mathcal{W}_E(\mathcal{A}', \mathrm{FinParity}(c))$ is non-empty, then $\Xi[\mathcal{A}']$ is non-empty.
3. Eve has a positional winning strategy from $\Xi[\mathcal{A}']$ in $(\mathcal{A}', \mathrm{FinParity}(c))$.

The proof is easy and follows the same lines as for the third step.

Theorem 1 gives the almost complete picture: the notable exception is the gap for finitary parity games, where we prove that $\ell + 1$ memory states are sufficient for Eve, yet without showing that any memory is required at all. Although we think that positional strategies always exist, we were not able to prove it. Our techniques through bounded parity games cannot be improved for this purpose, as we showed that for these games memory is required for Eve's winning strategies.

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:

- $(p, a, \mathrm{push}(b), q)$: allowed if the top stack element is $a \in \Gamma_\perp$, the symbol $b \in \Gamma$ is pushed onto the stack.
- $(p, \mathrm{pop}(a), q)$: 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 $(p, a, \mathrm{push}(b), q) \in \Delta$, then $((p, aw\perp), (q, baw\perp)) \in E$, for all words w in Γ^*.

Conditions. The coloring functions for parity conditions over 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 two examples witnessing interesting phenomena of pushdown finitary games (hence a fortiori of pushdown ωB games).

Example 6. Figure 8 presents a pushdown finitary Büchi game, where Eve wins for the bound 0, but loses the bounded uniform condition for any bound. 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. In other words, each visit of the two bottom states allows Adam to announce a number N and to prove that he can ensure a sequence of N consecutive non-Büchi states. 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.

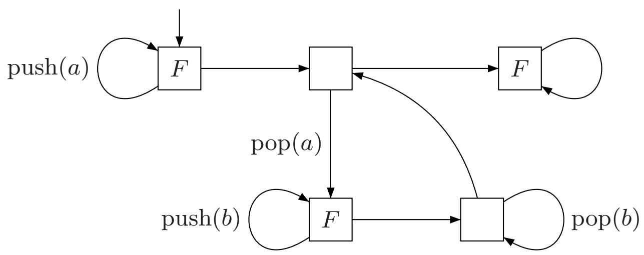

Figure 8. A pushdown game where Eve wins Büchi$(F, 0)$ but loses for any condition BndBüchi(F, N).

Example 7. Figure 9 presents a pushdown finitary Büchi game where Eve wins for the bound 2, but to do this she has to maintain a small stack. A play in this game divides into infinitely many rounds, which start by a visit to q. As in the previous example, each letter a on the stack is a "credit". A round consists in the following actions: first Eve chooses whether she wants to pop some a's from the stack (self-loop around q), and then moves the token to the Büchi state, second Adam decides either to push an a and start the next round or to go to the rightmost state p to pop some a's. The latter action should be understood as using credits (a's on the stack) to remain away from the Büchi state; using N credits, he can stay in p for N steps. It follows that Eve should everytime keep the stack low to avoid long stays in p. This rules out the greedy (attractor) strategy for her which would rush to the Büchi state without considering the stack; a wiser strategy ensuring the bound 2 is to start every round by popping the a pushed during the previous round.

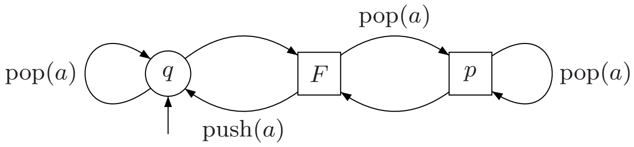

Figure 9. A pushdown game with finitary Büchi conditions where Eve has to maintain a small stack.

4.1 Regular sets of configurations and alternating $\mathcal{P}$-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 $\mathcal{P} = (Q, \Gamma, \Delta)$ is a classical alternating automaton over finite words: S is a finite set of control states, $\delta : S \times \Gamma \to \mathbb{B}^+(S)$ is the transition function (where $\mathbb{B}^+(S)$ is the set of positive boolean formulae over S), and F is a 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 using $q \in Q \subseteq S$ as initial state, with the standard alternating semantics. A set of configuration is said regular if it is accepted by an alternating $\mathcal{P}$-automaton.

The following theorem states that for very general conditions, the winning region is regular [34,35].

Theorem 3 ([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 $|Q|$.*

4.2 The collapse result

We denote LimitBounded(N) the set of plays which contain a suffix for which the counters are bounded by N. Note that unlike Bounded(N), the condition LimitBounded(N) is prefix-independent, so Theorem 3 applies.

Theorem 4 (The forgetful property). *For all pushdown ωB games, the following are equivalent:*

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

We refer to this result as a collapse result, as it shows that the non-uniform quantification (with respect to bound) of pushdown ωB games collapses to a uniform quantification (but using a slightly different bounding condition). It follows that we can associate to a pushdown ωB game a bound N, called the *collapse bound*, which only depends on the pushdown arena and the condition attached. Later in this section, we will show doubly-exponential lower bounds on this collapse bound.

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 6 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. Slightly modified, it also shows that Theorem 4 does not hold if LimitBounded is replaced by Bounded.

For the sake of readability, we abbreviate $\mathcal{W}_E(\mathrm{LimitBounded}(N) \cap \mathrm{Parity}(c))$ by $\mathcal{W}_E(N)$, and similarly $\mathcal{W}_E(\mathrm{Bounded} \cap \mathrm{Parity}(c))$ 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 3. 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 fool the bound N, but also $N+1$, and so on, yet remaining there. Iterating such strategies ensures to spoil the ωB-condition, which concludes the proof.

Remark 2. 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 are equal, *i.e* $\mathcal{B}_N = \mathcal{B}_{N+1}$, does not imply that from there on the sequence is constant. It follows that N can be *a priori* arbitrarily large.

We will later present examples showing that the bound N is at least doubly-exponential in the number of vertices, and exponential in the stack alphabet.

4.3 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 1) the collapse result (Theorem 4), and [7].
– Second, we prove the decidability of solving pushdown ωB games, generalizing the first item. This relies on the collapse result (Theorem 4) and [6].

We begin by proving the decidability of pushdown finitary games. Note that the second property in Theorem 4, namely:

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

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]. More specifically, let $\mathcal{G} = (\mathcal{A}, \mathrm{FinParity}(c))$ be a pushdown finitary parity game. We construct the memory structure $\mathcal{M}$ as in Proposition 4, which keeps track of the most important request. The arena $\mathcal{A} \times \mathcal{M}$ is again a pushdown arena, so it can be MSO-interpreted into the infinite binary tree. Now thanks to Theorem 1, Eve has a memoryless winning strategy in $\mathcal{G} \times \mathcal{M} = (\mathcal{A} \times \mathcal{M}, \mathrm{FinParity}(c))$ from her winning set. Such a strategy can be described as a set of edges, hence as a monadic second-order variable in an MSO formula over the infinite binary tree. Consider the following formula:

$$\exists X, \exists Y, \begin{cases} X \text{ represents a positional strategy} & \wedge \\ \text{all infinite paths end up in } Y & \wedge \\ \mathbb{B}Z,\ Z \text{ path in } X \cap Y \wedge \text{ the minimal color in } Z \text{ is odd} & \end{cases}$$

It expresses the existence of a positional strategy (X), a subset of vertices (Y) and a bound $N \in \mathbb{N}$ such that all plays consistent with X eventually enter in Y, where every request in answered within N steps. This is an existential bounding formula equivalent to the above property, whose satisfiability is decidable [7].

The second proof relies on [6], which studies two-way alternating parity cost-automata over infinite trees. For our purpose, we consider such automata over Γ-trees, which are infinite trees where each node has one child for each element in Γ. (Later, Γ will be the stack alphabet of a pushdown system.) We denote by $\mathrm{Act}(\Gamma)$ the following set of actions on a Γ-tree: $\{\mathrm{push}(a) \mid a \in \Gamma\} \cup \{\mathrm{pop}, \mathrm{skip}\}$, where push($a$) should be understood as "going down in the direction a", pop as "going up" and skip as "no move". (Note that they are in one-to-one correspondence with actions on a stack over Γ.)

Definition 1. *A two-way alternating automaton over Γ-trees (with k counters) is a tuple $\mathcal{B} = (Q, A, \delta, q_0, c)$, where Q is a finite set of states, A is a finite alphabet, $\delta : Q \times A \to \mathbb{B}^+(\mathrm{Act}(\Gamma) \times Q \times \{\varepsilon, i, r\}^k)$ is a transition relation (note that the counter actions appear here), q_0 is an initial state and $c : Q \to \mathbb{N}$ is a coloring function.*

Let $\mathcal{B}$ be such an automaton, it can be considered under two semantics: as an ωB-automaton or as a parity cost-automaton. Let t be a Γ-tree; the automaton $\mathcal{B}$ and the tree t induce an infinite-state game $(\mathcal{A}_{\mathcal{B},t}, \mathrm{Bounded} \cap \mathrm{Parity}(c))$. As an ωB-automaton, t is accepted by $\mathcal{B}$ if Eve wins the ωB game, and as a parity cost-automaton, t is accepted by $\mathcal{B}$ if there exists $N \in \mathbb{N}$ such that Eve wins the ωB game for the bound N.

The membership problem for such automata is a decision problem which asks, given a two-way alternating parity cost-automaton $\mathcal{B}$ and a regular tree t, whether t is accepted by $\mathcal{B}$. The following result is a consequence of [6]:

Theorem 5 ([6]). *The membership problem for two-way alternating parity cost-automata over regular trees is decidable.*

Indeed, they prove that two-way alternating parity cost-automata can be effectively translated into one-way alternating parity cost-automata, for which the membership problem is known to be decidable. We reduce the problem of solving a pushdown ωB game to the membership problem for two-way alternating parity cost-automata over a given regular tree.

Following [28], we first reduce the problem of determining the winner in a pushdown game to the membership problem for two-way alternating ωB automata over regular trees.

Consider the pushdown ωB game $\mathcal{G} = (\mathcal{A}, \mathrm{Bounded} \cap \mathrm{Parity}(c))$, and fix an initial configuration $(q_0, \bot)$ for the game. We define a two-way alternating ωB automaton: let $\mathcal{B} = (Q, \Gamma_\bot, \delta, q_0, c)$, where the transition relation δ is defined as follows: $\delta(p, a)$ is the disjunction of all possible transitions from (p, a) if $p \in Q_E$, and the conjunction of all possible transitions from (p, a) if $p \in Q_A$.

We run the automaton $\mathcal{B}$ on the Γ-tree t, which represents the stack contents: the label of the node u is the last letter of u if $u \neq \varepsilon$, and $\bot$ otherwise.

Lemma 4 ([28]). *The following are equivalent:*

- *Eve wins $\mathcal{G}$ from $(q_0, \bot)$.*
- *t is accepted by $\mathcal{B}$.*
- *$\exists \sigma$ strategy for Eve, $\forall \pi$ plays, $\exists N \in \mathbb{N}, \pi \in \mathrm{LimitBounded}(N) \cap \mathrm{Parity}(c)$.*

Now Theorem 4 implies that this is also equivalent to:

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

We construct a two-way alternating cost-automaton $\mathcal{B}'$ from $\mathcal{B}$ such that $\mathcal{B}'$ accepts t if and only if $\mathcal{B}$ accepts t (as an ωB-automaton). The automaton $\mathcal{B}'$ is obtained by adding at each transition 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 $\mathrm{LimitBounded}(N)$ and $\mathrm{Bounded}(N)$.

The main result of this section follows:

Theorem 6. *Solving a pushdown ωB-game is decidable.*

4.4 Lower bound on the collapse for finitary conditions

In this subsection, we prove lower bounds on the collapse bound which appears in Theorem 4, focusing on pushdown finitary Büchi games; note that this implies the same lower bound for the more general case of pushdown ωB games.

For the special case of pushdown finitary Büchi games, Theorem 4 can be stated as follows:

Corollary 1. *For all pushdown finitary Büchi games, there exists N such that:*

$$\mathcal{W}_E(\text{FinBüchi}(F)) = \mathcal{W}_E(\text{Büchi}(F, N)) \ .$$

The collapse bound depends on the following two relevant parameters of the pushdown arena: $n = |Q|$, the number of states, and $k = |\Gamma|$, the size of the stack alphabet.

We show that the collapse bound is at least doubly-exponential in the number of states and exponential in the stack alphabet.

The collapse bound for deterministic pushdown systems We start by considering deterministic pushdown systems, which is the very restricted case of pushdown games where from every configuration, there is only one transition, so no player has choice.

Standard pumping arguments shows that the collapse bound is at most exponential in both the number of states and the stack alphabet.

Lemma 5. *For all deterministic pushdown systems, we have:*

$$\mathcal{W}_E(\text{FinBüchi}(F)) = \mathcal{W}_E(\text{Büchi}(F, N)) \ ,$$

for $N = n^2 \cdot k^{n \cdot k+1}$.

Proof. We prove the left-to-right inclusion. Consider a path π, and assume it satisfies the finitary Büchi condition FinBüchi(F). We will show that it also satisfies the uniform Büchi condition for the bound $N = n^2 \cdot k^{n \cdot k+1}$. This collapse result is similar in fashion to the one obtained from the study of finitary games over finite arenas. It is clear that in this setting, if a path in a *deterministic* arena satisfies the finitary Büchi condition, then it satisfies the uniform Büchi condition for the bound $n-1$ (n being the number of vertices). Indeed, such a path is ultimately periodic, and the simple cycle it describes has length at most n. The content of this proof is to exhibit such a periodic pattern in π. Using a case distinction, we prove that either π ultimately repeats a cycle of length at most $n \cdot k^{n \cdot k}$, or ultimately repeats a cycle of increasing height (with respect to the stack) of length at most $n^2 \cdot k^{n \cdot k+1}$. The two cases we consider are the following, they are illustrated in Figure 10:

1. there is some configuration that appears twice;
2. no configuration appears twice.

Before going through these two cases, we state an observation that will be used several times in the proof: a simple path (that is, where each configuration appears at most once) whose maximal stack height difference is less than H has length at most $n \cdot k^H$.

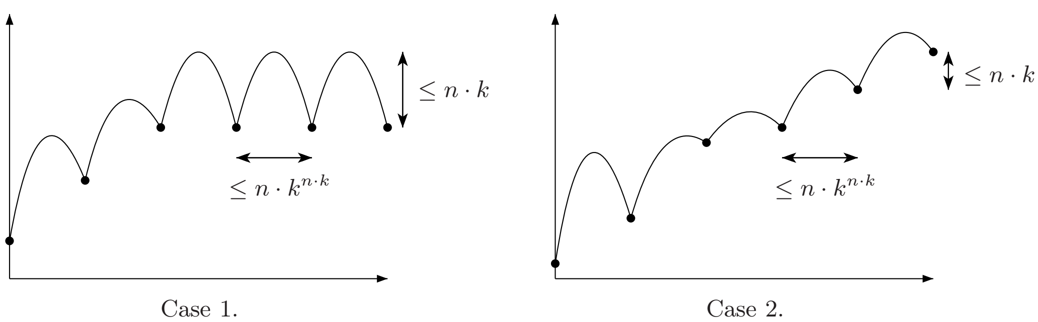

Figure 10. The case distinction for Lemma 5.

We start with the first case. It is clear that π is ultimately periodic; let C be the simple cycle described by π. We can see that the maximal stack height difference in the cycle C is less than $n \cdot k$, relying on a vertical pumping argument. It follows, relying on the earlier observation, that the cycle has length at most $n \cdot k^{n \cdot k}$.

We now focus on the second case. Here π is not ultimately periodic, but we will show that it repeats a cycle of increasing height. Define a step to be a configuration $(q, u\bot)$ in π whose stack height is minimal among the configurations that are visited after $(q, u\bot)$ in π. Since no configuration appears twice, it is clear that π has infinitely many steps. We say that two steps are consecutive in π if there are no steps inbetween in π. We first observe that two consecutive steps are separated by at most $n \cdot k^{n \cdot k}$ transitions: indeed the stack height, which remains higher than the height of the first step, must remain within the $n \cdot k$ intervall above the first step. Consider now the $n \cdot k$ first steps; two of them share the same state and top stack content, let us denote them $(q, au\bot)$ and $(q, avau\bot)$. The path π ultimately repeats a cycle of increasing height, as follows:

$$(q, au\bot) \to (q, avau\bot) \to \ldots \to (q, (av)^p au\bot),$$

whose length is bounded by $n^2 \cdot k^{n \cdot k+1}$. This concludes. ∎

The collapse bound proved in this lemma seems a priori quite large for such an easy case, as it is exponential in both n and k. However, Example 8 shows that it is asymptotically tight.

Example 8. Figure 11 presents a deterministic pushdown system, where the only path from $(F, \bot)$ satisfies the condition Büchi(F, N) for $N = O(2^n)$ but not for asymptotically less. This system encodes a number in binary in the stack with the least significant bit on the top of the stack. It has two phases: an initialization phase and an increment phase.

The initialization phase has n states and consists in pushing n times the symbol 0. The increment phase consists in adding one to the number encoded in the stack, *i.e* $1^k 0 u\bot \xrightarrow{*} 0^k 1 u\bot$. This phase goes on until it reaches the stack content $1^n\bot$, which is emptied to reach the only Büchi state F, and restarts from scratch. This pushdown process has $O(n)$ states and the collapse bound is $O(2^n)$.

An easy generalization consists in encoding in base k instead of 2, which would give an arena of size $O(k \cdot n)$ and a collapse bound asymptotically in k^n, *i.e* exponential in the number of states but not in the stack alphabet.

To obtain an arena where the collapse bound is exponential in both parameters, we perform slight modifications, as follows. In the latter arena, the numbers are encoded with n bits; we improve this by encoding the numbers using $k \cdot n$ bits. The increment phase remains the same. The initialization phase is not optimal; an ideal initialization phase would use $O(n)$ states to push $0^{k \cdot n}$ on the stack, but this is not possible, so we use a weaker initialization phase with n states that pushes:

$$\underbrace{(k-1)\ldots(k-1)}_{n} \cdot \underbrace{(k-2)\ldots(k-2)}_{n} \cdot \ldots \cdot \underbrace{1\ldots1}_{n} \cdot \underbrace{0\ldots0}_{n} \ .$$

The modified gadget is represented in Figure 12.

Since the counter does not start from 0 but from the number encoded in the latter stack, this new arena performs at bit less than $k^{k \cdot n}$ increment phases, but more than half this number, so its collapse bound is $O(k^{k \cdot n})$, exponential in both n and k.

The collapse bound for pushdown games For the following three examples, we denote by $p_1, p_2, \ldots$ the sequence of prime numbers, and by q_n the product of the first n prime numbers. We first start with the case where the stack alphabet has size one, *i.e* the subclass of one-counter pushdown games. Example 9 shows that in this case the bound is exponential in the number of states.

Example 9. Figure 13 presents a one-counter pushdown game, where for the condition Büchi(F, N), Eve wins for $N = q_n$ but not for $N - 1$. Eve first pushes a sequence of a's on the stack, then Adam chooses a prime number up to p_n and checks that the size of this sequence is divisible by this number. For this, Adam goes

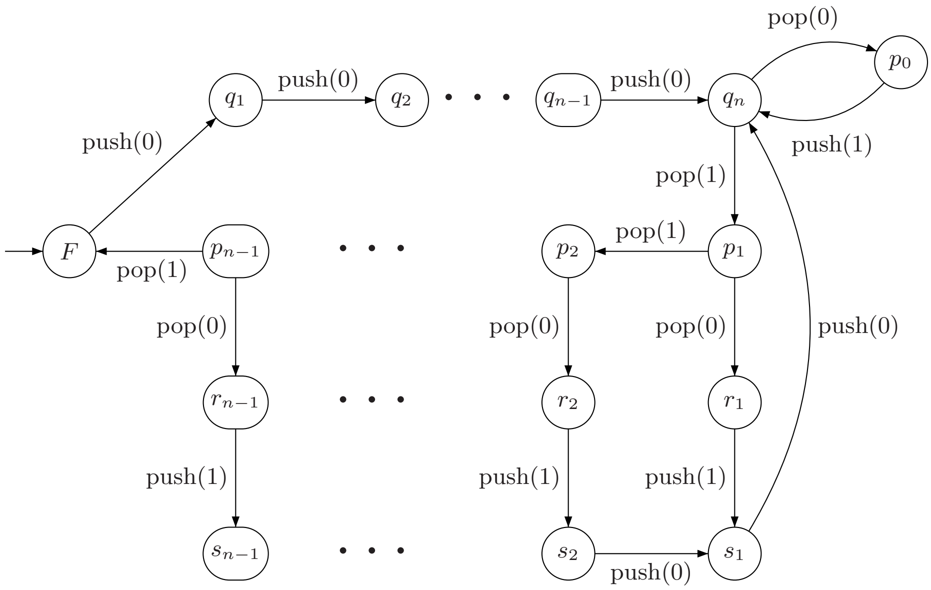

Figure 11. A deterministic pushdown system with an exponential collapse bound.

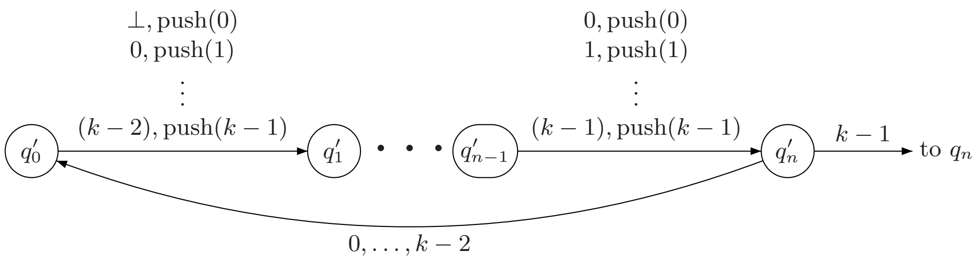

Figure 12. The improved initialization gadget.

to a loop of size p_k, going deterministically through it while popping one a at a time. If the empty stack is encountered in the beginning of the loop, then the size of the stack is divisible by p_k, and the game starts from scratch, visiting a Büchi state on the way.

Since Eve does not know in advance which prime number Adam is going to choose among $p_1, \ldots, p_n$, she has to push a non-empty sequence of size divisible by $q_n = \Pi_{1 \leq i \leq n} p_i$. The size of the arena is $O(\sum_{1 \leq i \leq n} p_i) = O(n \cdot p_n)$, whereas the smallest bound Eve can secure is q_n. An easy calculation shows that q_n is exponential in $O(n \cdot p_n)$.

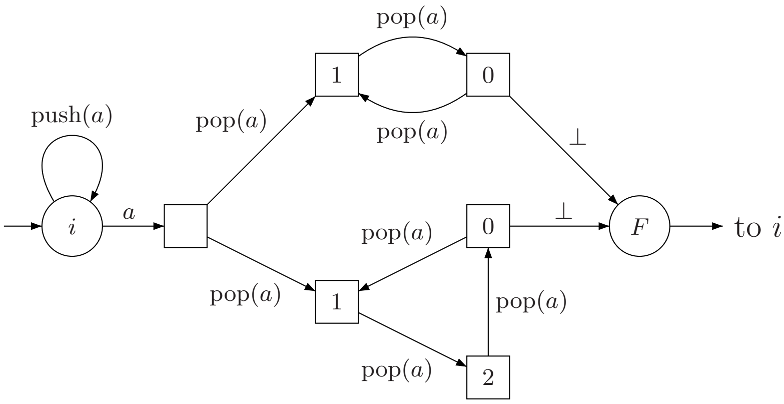

Figure 13. A one-counter pushdown game with an exponential bound.

We now consider a stack alphabet of size two, and combine the two ideas underlying Example 8 and Example 9, that is:

- Eve needs to push a sequence of exponential size;
- this sequence, seen as a binary decomposition of the number 0, is incremented by one until it reaches the sequence of only 1's, where the game empties the stack, starts from scratch and visits a Büchi state along the way.

Example 10 implements this idea, showing that the collapse bound is at least doubly-exponential in the number of states.

Example 10. Figure 14 presents a pushdown game, where Eve wins for the condition Büchi(F, N) for $N = O(2^{q_n})$, but not for $N = o(2^{q_n})$. In the figure, "sh" stands for stack-height: we saw in Example 9 how Adam can check that the size of the stack is a multiple of q_n, product of the n first prime numbers, using only $O(n \cdot p_n)$ states. As in the previous example, Eve first pushes a sequence of 0's on the stack, whose length must be a multiple of q_n, otherwise Adam wins by checking it. From s starts a binary increment similar to the one presented in Example 8; however in this example, the number of bits allowed was linear in the size of the arena, and we are now lifting this up to an exponential number of bits. So, we have to rely on the players' interactions to ensure that the binary increment is correctly executed. The action performed in the stack should be:

$$(s, 1^k 0 u \bot) \xrightarrow{*} (s, 0^k 1 u \bot).$$

The first part is deterministic:

$$(s, 1^k 0 u \bot) \xrightarrow{*} (c, 1 u \bot).$$

From c, Eve pushes some 0 on the stack. If she pushes less than k symbols, then Adam wins by checking, so she has to push at least k. Note, however, that she could push k plus any multiple of q_n, but she would only do herself a disservice.

The arena has size $O(n \cdot p_n)$, so N is doubly-exponential in the number of states.

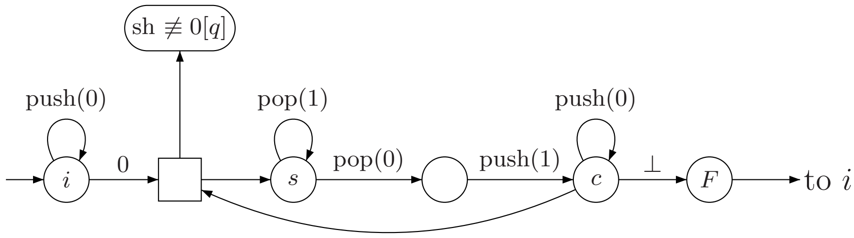

Figure 14. A pushdown game with a doubly exponential bound.

We now turn to a stack alphabet of size $2k+1$, and roughly "nest" Example 10.

Let $\Gamma = \{a_1, b_1, \ldots, a_k, b_k\} \cup \{\sharp\}$. The stack configurations we consider belong to the regular language:

$$L = \bigcup_{1 \le i \le k} \left(\{a_i, b_i\}^{q_n}\right)^+ \cdot \sharp \cdot \left(\{a_{i-1}, b_{i-1}\}^{q_n}\right)^+ \cdot \sharp \ldots \sharp \cdot \left(\{a_1, b_1\}^{q_n}\right)^+ .$$

Each block $\left(\{a_i, b_i\}^{q_n}\right)^+$ is seen as a number encoded in binary, where a_i is 0 and b_i is 1, which is initialized to $a_i^{q_n}$ and incremented by one step by step. However, the incrementation policy requires that to increment in the i^{th} block for $i < k$, one must increment in the $(i+1)^{\text{th}}$ block. Hence two increment phases in the i^{th} block are separated by 2^{q_n} increment phases in the $(i+1)^{\text{th}}$ block, which implies that two increment phases in the first block are separated by $2^{(k-1) \cdot q_n}$ transitions. Hence the 2^{q_n} increment phases required in the first block are executed within $2^{k \cdot q_n}$ steps. Example 11 constructs such a game.

Example 11. We sketch the construction of a pushdown game, where Eve wins Büchi(F, N) for $N = O(2^{k \cdot q_n})$, but not for asymptotically less.

First, following an easy adaptation of Example 9 we construct a game where Eve wins if and only if the stack content belongs to the language L. It has k components, each in charge of checking a block $\{a_i, b_i\}^{q_n}$. Eve first chooses i, and then Adam chooses a prime number to check that the size of the block is a multiple of the chosen prime number. Once a $\sharp$ symbol is reached, it is popped and the run goes on with the $(i-1)^{\text{th}}$ component, until the stack is empty. The size of this game is $O(k \cdot n \cdot p_n)$.

As before, Eve first pushes a sequence of a_1's on the stack, whose length must be a multiple of q_n, otherwise Adam wins by checking it. If he sends the pebble to d, then Eve chooses an i and starts a binary increment from v_i, similar to the one presented in Example 10. There are some differences, which appear at the end of an increment phase. If the block contained no a_i's, then the following case distinction occurs:

– If $1 < i \le k$, then the symbol $\sharp$ is popped from the stack, and another increment phase starts from v_{i-1}.
– If $i = 1$, then the game starts from scratch after paying a visit to a Büchi state.

Otherwise, the first a_i is turned into a b_i, and then Eve pushes some a_i's before sending the pebble to a state controlled by Adam. There, he can check that the stack content belongs to L, but he also has another option, following the case distinction:

– If $1 \le i < k$, then Adam can send the pebble back to the initial state, pushing a $\sharp$ symbol along the way.
– If $i = k$, then Adam can send the pebble to v_k.

Whenever Adam sends the pebble back to the initial state after an increment phase of the i^{th} block, Eve has no choice but to push a sequence of a_{i+1}'s on the stack, whose length must be a multiple of q_n, otherwise Adam wins since the stack content would not belong to L.

The arena obtained has size $O(k \cdot n \cdot p_n) + O(k) = O(k \cdot n \cdot p_n)$, so the bound required for Eve to win the uniform Büchi condition is doubly-exponential in the number of states and exponential in the stack alphabet.

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 [10,23]. 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:

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

5.1 A reduction from finitary parity to bounded parity

The reduction relies on a *restart* gadget. We consider a pushdown game with finitary parity conditions, given by the coloring function $c : Q \to [d]$, where we assume d to be odd. Between every edge of the game we add a restart gadget, where Eve can choose either to follow the edge, or to "restart": this entails that first a vertex with priority 0 is visited, where Adam can stay as long as he wants by pushing on the stack a new symbol $\sharp$, and then Eve takes over, staying in a vertex with priority d until all the $\sharp$ symbols are popped away from the stack, before following the original edge. 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 let a request unanswered for a long time. Therefore, Eve can restart only finitely many times. The gadget is represented in Figure 15.

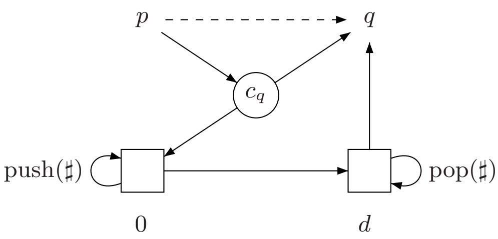

Figure 15. The restart gadget.

Lemma 6. *Eve wins the finitary parity game if and only if she wins the reduced bounded parity game.*

Proof. We prove both implications.

– Assume Eve wins the finitary parity game, and let σ be a winning strategy. We construct a strategy σ_R in the reduced bounded parity game. It maintains a counter, initially set to 1, whose value is denoted by N. The strategy σ_R plays consistently with σ. It restarts if there exists a request made before the last N transitions that has not been serviced, and if so increments the counter by one. We argue that σ_R is winning for the bounded parity condition. Consider π_R a play consistent with σ_R: if it remains in the restart gadget forever (Adam pushes $\sharp$ forever), it is winning. Otherwise, if a restart occurs for a value N of the counter, then there is a pending request not serviced within N transitions, which got serviced through the restart. Let π be the corresponding play in the parity game, where we skip the restarts: π is consistent with σ, so it satisfies the finitary parity condition. Now, it is clear that π_R contains only finitely many restarts, otherwise it would include requests that are not serviced within N transitions, for arbitrary N, which contradicts the fact that π satisfies the finitary parity condition. It follows that π_R and π coincide from some point onwards, so π_R satisfies the bounded parity condition, and σ_R is a winning strategy in the reduced bounded parity game.

– Conversely, assume that Adam wins the finitary parity game, and let τ be a winning strategy. We construct a strategy τ_R in the reduced bounded parity game. As for the case of Eve, it features a counter, initialized to 1 and whose value is denoted by N. Outside the restart gadget, τ_R plays consistently with τ, and inside the restart gadget, τ_R pushes exactly N times the symbol $\sharp$, and then increments the counter by one. Consider π_R a play consistent with τ_R, there are two cases: either it includes finitely many uses of the restart gadgets, or infinitely many. In the first case, π_R coincides from some point onwards with a play π consistent with τ, so it spoils the bounded parity condition. In the second case, the request made in the last vertex of the restart gadget remains unserviced for an unbounded time, so the bounded parity condition is fooled as well. It follows that π_R spoils the bounded parity condition, thus τ_R is a winning strategy in the reduced bounded parity game.

∎

5.2 The special case of Büchi conditions

In the study of finitary games over finite graphs [14], 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, the case of finitary Büchi specializes again and collapses to Büchi.

Lemma 7. *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 $\text{FinBüchi}(F) \subset \text{Büchi}(F)$. The converse inclusion follows from memoryless determinacy for the condition $\text{Büchi}(F) \cap \text{BndSt}$ [10]: assume 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 $\text{FinBüchi}(F)$.

Note however that in general, for a pushdown game, $\mathcal{W}_E(\text{FinParity}(c) \cap \text{BndSt}) \neq \mathcal{W}_E(\text{Parity}(c) \cap \text{BndSt})$.

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 16.

$$\begin{tikzcd}
\begin{array}{c}\text{FinParity}(c)\\ \text{and}\\ \text{BndSt}\end{array} \arrow[r, bend left, "\text{restart}"] & \begin{array}{c}\text{BndParity}(c)\\ \text{and}\\ \text{BndSt}(Q)\end{array} \arrow[r, bend left, "\times\mathcal{M}"] & \begin{array}{c}\text{FinBüchi}(F)\\ \text{and}\\ \text{BndSt}(Q)\end{array} \arrow[r, bend left, "\text{collapse}"] & \begin{array}{c}\text{Büchi}(F)\\ \text{and}\\ \text{BndSt}(Q)\end{array}
\end{tikzcd}$$

Figure 16. 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 the reduction from bounded parity to finitary Büchi by composing with the memory structure from Proposition 4, keeping track of the most urgent pending request. We are now left with a 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 7 deals with stack boundedness, not restricted to a subset Q. 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 can be used only finitely many times.

Formally, we first need to extend the memoryless determinacy for the condition Büchi and restricted stack boundedness.

Lemma 8. *For all pushdown games with condition Büchi and restricted stack boundedness, Eve has a memoryless winning strategy from her winning set.*

Proof. The proof is a straightforward adaptation of Proposition 1 from [23]. ∎

Now, consider σ a memoryless strategy ensuring the condition Büchi and restricted stack boundedness in the pushdown game obtained through the above reductions; we prove that σ ensures finitary Büchi. Let π be a play consistent with σ, there are two cases: either the play remains forever in the restart gadget, or from some point onwards the restart gadget is not used anymore. In the first case, the finitary Büchi condition is clearly satisfied. In the other case, the play satisfies the general stack boundedness condition, and the same reasoning as for Lemma 7 concludes that the finitary Büchi condition is satisfied.

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 [10,23] 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 7. *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 [22]. 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 LICS anonymous reviewers for their valuable comments.

References

1. Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Monotonic and downward closed games. *J. Log. Comput.*, 18(1):153–169, 2008.
2. Bowen Alpern and Fred B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985.
3. Rajeev Alur and Thomas A. Henzinger. Finitary fairness. *ACM Trans. Program. Lang. Syst.*, 20(6):1171–1194, 1998.
4. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
5. Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Modular strategies for recursive game graphs. *Theor. Comput. Sci.*, 354(2):230–249, 2006.

6. Achim Blumensath, Thomas Colcombet, Denis Kuperberg, and Michael Vanden Boom, 2013. Personal communication.
7. Mikołaj Bojańczyk. A bounding quantifier. In *CSL*, pages 41–55, 2004.
8. Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296, 2006.
9. Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In *STACS*, pages 648–660, 2012.
10. Alexis-Julien Bouquet, Olivier Serre, and Igor Walukiewicz. Pushdown games with unboundedness and regular conditions. In *FSTTCS*, pages 88–99, 2003.
11. Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.
12. J. Richard Büchi and Lawrence H. Landweber. Definability in the monadic second-order theory of successor. *J. Symb. Log.*, 34(2):166–170, 1969.
13. Krishnendu Chatterjee and Nathanaël Fijalkow. Finitary languages. In *LATA*, pages 216–226, 2011.
14. Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009.
15. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, pages 139–150, 2009.
16. Thomas Colcombet. Fonctions régulières de coût. Habilitation Thesis, in French, 2013.
17. Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy. Regular temporal cost functions. In *ICALP (2)*, pages 563–574, 2010.
18. Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.
19. Thomas Colcombet and Christof Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79, 2010.
20. E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *FOCS*, pages 328–337, 1988.
21. E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377, 1991.
22. Nathanaël Fijalkow and Martin Zimmermann. Cost-parity and cost-streett games. In *FSTTCS*, pages 124–135, 2012.
23. Hugo Gimbert. Parity and exploration games on infinite graphs. In *CSL*, pages 56–70, 2004.
24. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
25. Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982.
26. Daniel Kirsten. Desert automata and the finite substitution problem. In *STACS*, pages 305–316, 2004.
27. Eryk Kopczyński. Half-positional determinacy of infinite games. In *ICALP (2)*, pages 336–347, 2006.
28. Orna Kupferman and Moshe Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *CAV*, pages 36–52, 2000.
29. Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.
30. Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993.
31. David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.
32. Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.
33. Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969.
34. Olivier Serre. Note on winning positions on pushdown games with ω-regular conditions. *Inf. Process. Lett.*, 85(6):285–291, 2003.
35. Olivier Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris 7 - Denis Diderot, 2006.
36. Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.
37. Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.
38. Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.