Faster Algorithms for Finitary Games*

Florian Horn

Liafa, Université Paris 7, Case 7014, 2 place Jussieu, F-75251 Paris 5, France
horn@liafa.jussieu.fr

Abstract. The theory of games is a prominent tool in the controller synthesis problem. The class of ω-regular games, in particular, offers a clear and robust model of specifications, and present an alternative vision of several logic-related problems. Each ω-regular condition can be expressed by a combination of safety and liveness conditions. An issue with the classical definition of liveness specifications is that there is no control over the time spent between two successive occurrences of the desired events. Finitary logics were defined to handle this problem, and recently, Chatterjee and Henzinger introduced games based on a finitary notion of liveness. They defined and studied finitary parity and Streett winning conditions. We present here faster algorithms for these games, as well as an improved upper bound on the memory needed by Eve in the Streett case.

1 Introduction

Games are one of the most practical tools to study the controller synthesis problem in open systems. The setting of the problem is translated into an arena, while the controller and the environment are the players that make decisions based on the current state of the system and the former actions of their opponent. The desired behaviour of the system is given as a constraint over the sequence of system states, usually in the form of an ω-regular condition [MP92]. The study of these ω-regular games is the subject of a very large part of the games theory (two out of many, [Tho95, AHK02]). These games also present the advantage of giving alternate tools to solve problems of model-checking and verification. However, they present some weaknesses when they are used in actual synthesis of controllers. Each ω-regular condition can be expressed by a combination of liveness and safety conditions. Safety specifications are sound in terms of controller synthesis: they ask for the controller to prevent the occurrence of undesirable events, as long as some other condition does not change. Liveness specifications, however, are not as satisfying. The classical definition asks only for the desired event to happen *eventually*, without any constraints on the number of transitions it may take. This allows more robust specifications, in the sense that they do not depend on the way a system is represented. In one-shot liveness (reachability), this is perfectly natural: the actual number of transitions

* Work supported by the EU-TMR network GAMES. Some of this work was done in RWTH, Aachen.

O. Grumberg and M. Huth (Eds.): TACAS 2007, LNCS 4424, pp. 472–484, 2007.

depends more on the particular representation we use than on the actual properties of the system studied. But as soon as we consider Büchi conditions, there exists behaviours compatible with these specifications in which the number of transitions between two visits to the target set is unbounded. On finite graphs, one can always take shortcuts to avoid these cases. When we consider parity conditions in open systems, however, there are cases where there is no bounded solution. Finitary conditions, in which the unbounded behaviours are forbidden were introduced in [AH94]. More recently, [BC06] proposed a logic based on a variant of the notion of ω-regularity that introduced bounds on the size of the set of states considered. A fragment of this logic, where the bounds concerns only the distance between events, express the finitary conditions.

In [CH06], Chatterjee and Henzinger introduced finitary games and studied the cases of parity and Streett specifications. They proved that both games were determined and provided algorithms computing the winning regions. The finitary parity problem was also proved to be in $\mathtt{NP} \cap \text{co-}\mathtt{NP}$. We present here faster algorithms for both games, using Turing reductions to other variations on these games. The finitary parity game problem is proved to be in $\mathtt{P}$, with a time complexity of $m \cdot n^2$, where n is the number of states in the arena and m is the number of edges. In comparison, the original algorithm of [CH06] had a time complexity of $O(n^{2c-3} \cdot c \cdot m)$ (c is the number of colors in the parity condition). The finitary Streett algorithm is faster than the original reduction to finitary parity with a complexity of $O(4^k \cdot k^2 \cdot m^2 \cdot n)$, where n and m still denotes the numbers of states and edges, and k is the number of pairs in the Streett condition. The algorithm of [CH06], based on a reduction to finitary parity games, had a complexity of $O((n \cdot k! \cdot k^2)^{2k-3} \cdot m \cdot k! \cdot k^3))$. In addition, our algorithm yields a strategy for Eve that uses $2^k \cdot k$ memory states, instead of $k! \cdot k^2$ in the strategy derived from the reduction.

Outline of the Paper. Section 2 defines the general notions on games we use in all the paper. Sections 3 defines several variants of parity games, including finitary ones, and gives algorithms that solve them. Section 4 does the same for Streett games. Finally, section 5 summarizes the results and presents some ideas about future work in this domain.

2 Definitions

A 2-player game is a tuple (V, E, Win) consisting of a graph (V, E) containing a token, and a winning condition $\mathit{Win} \subseteq V^\omega$. The token is always in one of the states and can only move along the edges. The set of states V is partitioned into Eve's states (V_E, represented by circles) and Adam's states (V_A, represented by squares). The owner of the state containing the token chooses the next state. An infinite play $\rho = q_1, q_2, \ldots$ is a sequence of states visited by the token, respecting the edge relation: $(q_i, q_{i+1}) \in E$ for all $i > 0$. We consider only infinite plays, by

assuming that every state has at least one successor. A play in *Win* is winning for Eve. Otherwise, it is winning for Adam. For complexity computations, we will always denote by n the total number of states, and by m the total number of edges.

In this paper, we will only consider games on finite graphs. Most of the notions presented in this section also exist on infinite graphs, but our algorithms are not adapted to those. We will now introduce several definitions and tools used to solve games. See [Tho95, Zie98] for more detailed proofs.

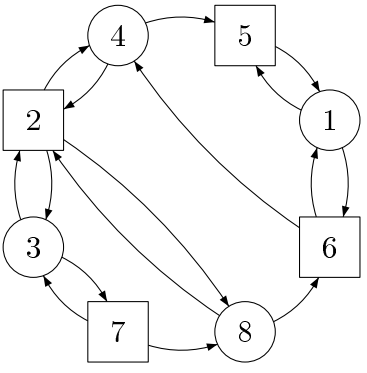


Fig. 1. A game graph

Definition 1. *A subgame of a game $G = (V, E, Win)$ is a game defined on a subset V' of V such that each state in V' has a successor in V'. The edges and the wining set are restrictions of E and Win to V'.*

The *arena* of a game is the graph (V, E), including the partition between V_A and V_E. A *sub-arena* of an arena is the arena of a subgame. Many notions about games depend only on the arena of the game, and this allows us to export them from a game to another, as long as they are played on the same arena. The central notion of play, in particular, depends only of the arena.

A strategy for Eve (resp. Adam) is a function σ from V^*V_E (resp. V^*V_A) to V such that for any finite prefix w and any state q, there is an edge between q and $\sigma(w.q)$. Informally, a strategy for player P is a method of extending any finite prefix ending in a state of P. A strategy is positional if σ depends only on the current state. It has a finite memory if it can be realized by a finite-state transducer. A play is consistent with a strategy σ for P if $\forall i \in \mathbb{N}, \rho_i \in V_P \Rightarrow \rho_{i+1} = \sigma(\rho_{1..i})$. All these notions depend only on the arena of the game.

A strategy for P is winning for P from a state q if each play starting in q and consistent with it is won by P. The winning region of a player P in a game G, denoted by $Win_P(G)$, is the set of states from where P has a winning strategy.

The *attractor of W for player P in the game G*, denoted $Attr_P^G(W)$, is the set of states from which P can ensure that the token will reach the set W in a finite number of moves. It is computed inductively as usual:

$$\begin{aligned} W_0 &= W \\ W_{n+1} &= W_n \\ &\quad \cup \{q \in V_P \mid \exists q' \in W_n, (q, q') \in E\} \\ &\quad \cup \{q \in V \mid \forall q', (q, q') \in E \Rightarrow q' \in W_n\} \end{aligned}$$

The *attractor strategy* for player P is positional, and consists in always going from a state of W_n to a state in W_{n-1}, thus getting closer to W. The complexity of the computation of either the attractor set or the attractor strategy is $O(m)$.

A *trap* is the dual of an attractor, and hence is a set from which one of the players cannot escape: A trap $T \subseteq V$ for player P is a region such that each state belonging to the other player has a successor in T, and each state in V_P has *all* its successors in T. Note that the complement of an attractor is a trap for the same player, and that a trap is always a sub-arena. Once again, the notions of attractor and trap depend only on the arena of the game.

In this paper, we will study the relations between several winning conditions defined on the same arena.

3 Parity Games

3.1 Parity Conditions

A parity coloring p is a function that associates an integer to each state of an arena $\mathcal{A}$. A parity arena is an arena equipped with a parity coloring. All the parity games that we define depend only on the parity arena they are played on. It is thus legitimate to talk about *the weak parity game on the arena* $\mathcal{A}_p$ without further precision. In complexity computations, we will denote the number of colors in a parity arena by c.

The study of parity games is usually concerns one of the two following kind:

Weak parity games: A play is winning for Eve if the least color appearing in the play is even.

(Classical) parity games: A play is winning if the least color appearing infinitely often in the play is even.

In this paper, we will study another kind of parity games, called finitary parity. These games were introduced by Chatterjee and Henzinger in [CH06]. Intuitively, a play is winning for Eve in finitary parity if for each odd color that occurs infinitely often, a smaller even color occurs infinitely often, as in classical parity, with the added constraint that the delay between an occurrence of an odd color and the next smaller even color must be ultimately bounded.

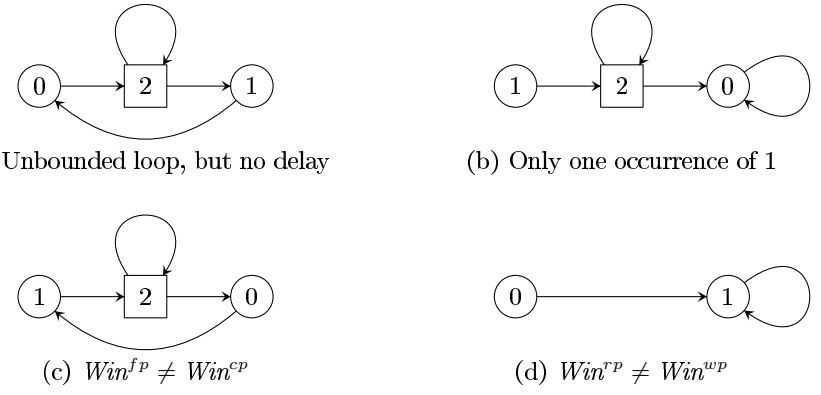
The formal definition uses the notion of delay sequence of a play:

Definition 2. *The delay sequence $d(\rho)$ of a play ρ on a parity arena $\mathcal{A}_p$ is defined as follows:*

- *If $p(\rho_i)$ is even, then $d(\rho)_i = 0$.*
- *If $p(\rho_i)$ is odd, then $d(\rho)_i$ is the smallest j such that $p(\rho_{i+j})$ is even and $p((\rho)_{i+j}) < p(\rho_i)$. Note that if there is no such j, $d(\rho)_i = \infty$.*

A play ρ on a parity arena $\mathcal{A}_p$ is winning for Eve in the finitary parity game if and only if $d(\rho)$ is ultimately bounded. Note that, as the delay function can take infinite values, *ultimately bounded* is a weaker property than simply *bounded*.

Figure 2 gives some examples of how these games work. In arena 2(a), Adam can control the time between occurrences of 1, but an occurrence of 0 always comes immediately after. The delay sequence is made only of 0's and 1's. Thus Eve wins in the finitary parity game. In arena 2(b), Adam can control the time



(a) Unbounded loop, but no delay

(b) Only one occurrence of 1

(c) $Win^{fp} \neq Win^{cp}$

(d) $Win^{rp} \neq Win^{wp}$

Fig. 2. Examples of parity games

spent between the first 1 and the next 0, or even choose to never go to 0. The first element of the sequence can thus be as high as Adam wants, or even infinite. But all following values will be equal to 0, so Eve wins in the finitary parity game. In the weak-parity game, Adam would have won if the play had begun in the state 1. In arena 2(c), however, Adam can delay the time between a 1 and the next 0 as long as he wants before allowing the loop to go on. Thus he can make the delay function unbounded and win the finitary game. Notice that he would not win in the classical parity game.

In [CH06], Chatterjee and Henzinger proved the following results about finitary parity games:

- Finitary parity games are determined ($Win_A^{fp}(\mathcal{A}) \cup Win_E^{fp}(\mathcal{A}) = \mathcal{A}$).
- In her winning region, Eve has a positional winning strategy.
- Adam may need strategies with infinite memory in order to win.
- Winning regions can be computed in time $O(n^{2c-3} \cdot c \cdot m)$.
- Deciding the winner in a given state is in NP $\cap$ co-NP.

Our algorithm for finitary games uses yet another kind of parity games, that we call repeating parity game. These games are also defined in terms of delay sequence: A play ρ is winning if the associated delay sequence only takes finite values. Intuitively, it means that for each occurrence of an odd color, there is later an occurrence of a smaller even color.

These games are different from the other parity games we defined. In figure 2(b), Adam will win if the play starts in state 1, by blocking the token in state 2, while Eve would have won a finitary or classical game. In figure 2(d), he wins again, even if the play starts in state 0, while Eve would have won a weak game. The definition does not supposes that the delay function is bounded. On finite arenas, however, it is easy to see that Eve can bound the delay function to n in her winning region : If she can reach a smaller even color, then she can reach it in less than n moves.

3.2 Algorithms

Another way of thinking about repeating parity games is to consider them as weak-parity games where Adam can reset the set of visited states whenever he wants, but will lose if he does so infinitely often. This intuition is formalized in the lemma 3.

Lemma 3. *α : The winning region of Adam in the weak-parity game on an arena $\mathcal{A}_p$ is also winning for him in the repeating parity game on the same arena. The attractor of this region is also winning for him in this game.*
β : If Eve wins everywhere in the weak-parity game on an arena $\mathcal{A}_p$, then she wins everywhere in the repeating parity game on the same arena.

Before we give the proof for this lemma, a short review of how the winning regions are computed in a weak-parity game is in order. The full algorithm comes from [LT00]. It works by removing attractors for each players alternatively. We consider that the smallest color is 0[1]. The region labeled by 0 is immediately winning for Eve, as any play starting from this region will have 0 as smallest occurring color. For the same reason, any state in Eve's attractor to this region will be winning for her. All these states are winning for Eve, and can now be removed from the game in order to compute the winning regions in the rest of the game: Eve cannot go to these states, and Adam will never want to go there, as Eve could force the token to visit a state labeled 0. The remainder of the arena is a (parity) sub-arena, as it is a trap, and its size is strictly smaller. We can thus use the same algorithm to get the winning regions of both players in this new game.

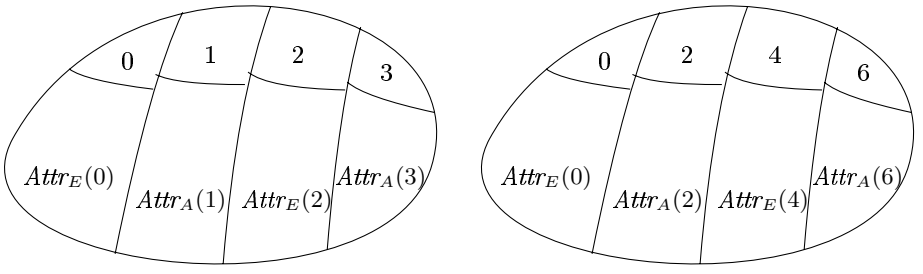
An illustration of how this algorithm works is given in figure 3(a). Notice that all the attractors and regions are relative to earlier computation: there could be 1's in the region $Attr_E(0)$, for example. However, there cannot be a 0 in the region $Attr_A(1)$, as it would belong to $Attr_E(0)$ which was defined earlier. In the same way, the attractors are computed relatively to the subgames. $Attr_A(1)$, for example, is the region where Adam can force the token to a state labeled 1 without crossing $Attr_E(0)$. Figure 3(b) shows the special case where Eve wins everywhere. It is computed in the same way, but the odd regions and attractor happens to be empty (which is fitting, since they are regions winning for Adam). Obviously, there could be odd colors in the arena, but each belongs to attractors of smaller even colors.

A naïve study of this algorithm leads to a worst-case time complexity of $O(c \cdot m)$. However, a careful use of data structures reduces this time to $O(m)$ [Cha06].

We will now give the proof of lemma 3.

Proof. α : A play is winning for Adam in the weak parity game if the smallest color visited is odd. Obviously, a smaller even color cannot occur later. Thus

[1] If this is not the case, just replace 0 by the smallest color present in the arena, and Eve by Adam this color is odd



(a) The weak-parity algorithm (b) Eve wins everywhere

Fig. 3. An arena where Eve wins everywhere in weak-parity

$Win_A^{wp}(\mathcal{A}) \subseteq Win_A^{rp}(\mathcal{A})$. The case of the attractor is solved by the following observation: If a play ρ is winning for Adam in the repeating parity game, then each play of the form $w \cdot \rho$ is also winning for him in this game. Thus $Attr_A(Win_A^{wp}(\mathcal{A})) \subseteq Win_A^{rp}(\mathcal{A})$. □

β : An arena where Eve wins everywhere in the weak-parity game looks like figure 3(b). In this game, Eve needs only to play according to the attractors' strategies whenever the token is not in one of the top-most regions. This guarantees that the token will get to an even state in the top-most regions without crossing a smaller odd color. Thus Eve also wins everywhere in the repeating parity game. On a finite arena with n states, using this strategy also guarantees that the delay between an odd color and the next smaller even color is at most n. □

This lemma leads directly to algorithm 1[2].

Algorithm 1. Algorithm computing the winning regions of Adam and Eve for the repeating parity game

Require: Algorithm for computing the winning regions in a weak-parity game
  input $\mathcal{A}, p$
  $\mathcal{B} \leftarrow \mathcal{A}$
  repeat
    $\mathcal{B} \leftarrow \mathcal{B} \setminus Attr_A(Win_A^{wp}(\mathcal{B}, p))$
  until $Win_A^{wp}(\mathcal{B}, p) = \emptyset$
  return $\mathcal{A} \setminus \mathcal{B}, \mathcal{B}$

The termination of the algorithm 1 is guaranteed by the fact that in each **repeat** loop, $\mathcal{B}$ loses at least one state. This limits the number of times the loop can be repeated to n. As weak-parity games are solved in time $O(m)$, the global complexity of our algorithm is $O(m \cdot n)$.

[2] In this algorithm, as in the others, the first component returned is Adam's winning region, while the second is Eve's winning region.

The validity of this algorithm derives directly from lemma 3.

We will now solve finitary games using the same kind of construction. Lemma 4 relates the winning regions of repeating parity and finitary parity, in a way very similar to lemma 3.

Lemma 4. *α : The winning region of Eve in the repeating parity game on an arena $\mathcal{A}_p$ is also winning for her in the finitary parity game on the same arena. The attractor of this region is also winning for her in this game.*
β : If Adam wins everywhere in the repeating parity game on an arena $\mathcal{A}_p$, then he wins everywhere in the finitary parity game on the same arena.

Proof. α : Eve's winning region in the repeating parity game is the region from where she can guarantee that the delay function remains finite. On this region, she can use the strategy described in the proof of lemma 3 which guarantees that the delay function will be bounded by n. The finitary parity game asks only for this function to be ultimately bounded. Thus $Win_E^{rp}(\mathcal{A}_p) \subseteq Win_E^{fp}(\mathcal{A}_p)$. As the finitary parity condition is prefix independent, we can conclude that $Attr_E(Win_E^{rp}(\mathcal{A}_p)) \subseteq Win_E^{fp}(\mathcal{A}_p)$ □

β : The second part of the proof is more complex. Adam has a strategy π that is winning everywhere in repeating parity. From it, we derive the following strategy π':

1. Set b to 1.
2. Play the strategy π with initial memory from the state where the token is now until there is a sequence with an odd priority followed by b moves without seeing a smaller even priority.
3. Increment b.
4. Go back to step 2.

It is immediate that if a play consistent with this strategy behaves in a way such that Adam goes infinitely often through the loop, then it is winning for Adam in the finitary parity game. The only point that could cause trouble is to get out of step 2. But a play that would get stuck in this state would be a play consistent with π where for each occurrence of an odd color, there is an occurrence of a smaller even color in the next b moves. This would be a contradiction to the hypothesis that π is winning for Adam in the repeating parity game. Thus π' is winning for Adam in the finitary parity game. □

As we did for repeating parity, we use this lemma to build Algorithm 2, computing the winning regions in finitary parity games.

As in Algorithm 1, the termination and complexity are guaranteed by the fact that the **repeat** loop removes one state from $\mathcal{B}$. Likewise, the complexity is n times the complexity of the former algorithm, or $O(m \cdot n^2)$. This complete the proof of the following theorem:

Theorem 5. *Deciding the winner of a state in a finitary parity game can be done in polynomial time. In particular, Algorithm 2 computes the winning regions of a game with n states and m edges in time $O(m \cdot n^2)$.*

To give a comparison, the algorithm of [CH06] was running in time $O(n^{2c-3} \cdot c \cdot m)$. Chaterjee and Henzinger also proved that the problem was in NP ∩ co-NP.

Algorithm 2. Algorithm computing the winning regions of Adam and Eve for the finitary parity game

Require: Algorithm for computing the winning regions in a repeating parity game
  input $\mathcal{A}, p$
  $\mathcal{B} \leftarrow \mathcal{A}$
  repeat
    $\mathcal{B} \leftarrow \mathcal{B} \setminus Attr_E(Win_E^{rp}(\mathcal{B}, p))$
  until $Win_E^{rp}(\mathcal{B}, p) = \emptyset$
  return $\mathcal{B}, \mathcal{A} \setminus \mathcal{B}$

4 Streett Games

4.1 Streett Conditions

A Streett coloring s over an arena $\mathcal{A}$ is a set of pairs of sets of states of $\mathcal{A}$. The first element of a pair is usually called a *request*, and the second element is the corresponding *response*. A Streett arena $\mathcal{A}_s$ is an arena equipped with a Streett coloring. The rank of a Streett arena is the number of pairs that constitute the Streett coloring. The rank of a Streett game is the rank of its arena. In complexity computation, the rank of the Streett condition will be denoted by k. As was the case for parity games, all the variants of Streett games that we will define depend only on the Streett arena they are played on. Again, there are two classical versions of the Streett games:

Weak Streett games: A play is winning for Eve if for each request that occurs in the play, the corresponding response also occurs.

(Classical) Streett games: A play is winning for Eve if for each request occurring infinitely often in the play, the corresponding response also occurs infinitely often.

Chatterjee and Henzinger also introduced a finitary version of the Streett games in [CH06]. Intuitively, a play is winning for Eve in finitary Streett if for each request that occurs infinitely often, the corresponding response occurs infinitely often, as in classical Streett, with the added constraint that the delay between an occurrence of a request and the next corresponding response must be ultimately bounded.

The formal definition also uses a notion of delay sequence derived from a play:

Definition 6. *The delay sequence $d(\rho)$ of a play ρ on a Streett arena $\mathcal{A}_p$ is defined as follows:*

- *If ρ_i does not belong to a request, then $d(\rho)_i = 0$.*
- *If ρ_i belongs to the request of only one pair, then $d(\rho)_i$ is the smallest j such that ρ_{i+j} belong to the corresponding response. Note that if there are no such j, $d(\rho)_i = \infty$.*
- *If ρ_i belong to several requests, then $d(\rho)_i$ is the maximum of the values computed with the method above for each request.*

A play ρ on a Streett arena $\mathcal{A}_p$ is winning for Eve in the finitary Streett game if and only if $d(\rho)$ is ultimately bounded.

In [CH06], Chatterjee and Henzinger proved the following results about finitary Streett games:

- Finitary Streett games are determined. ($Win_A^{fs}(\mathcal{A}) \cup Win_E^{fs}(\mathcal{A}) = \mathcal{A}$)
- In her winning region, Eve has a strategy that uses no more than $k! \cdot k^2$ memory states.
- Adam may need strategies with infinite memory in order to win.
- Winning regions can be computed in time $O((n \cdot k! \cdot k^2)^{2k-3} \cdot m \cdot k! \cdot k^3)$.

As for parity games, we will use another kind of Streett games in our algorithm, called request-response games. These games will have the same place in the Streett algorithm than repeating parity had in the parity algorithm. However, there are two significant differences: there is no relation[3] between these games and weak-parity games, and they were defined and studied by Wallmeier, Thomas and Hutten in [WHT03]. Even if these games do not bear the name Streett, they are defined by a Streett arena:

A play is winning in a request-response game if its delay sequence takes only finite values, *i.e.* if for each occurrence of a request, there is later an occurrence of a corresponding response.

Wallmeier et al. present an algorithm to solve request-response games in [WHT03]. It is based on a reduction to generalized Büchi games. The time complexity of their algorithm is $O(4^k \cdot k^2 \cdot m^2)$. The strategy for Eve that is derived from this algorithm has the property that in each play consistent with it, each request is matched by a corresponding response in the next $k \cdot n$ moves.

The following lemma relates finitary Streett games and request-response games:

Lemma 7. *α : The winning region of Eve in the request-response game on an arena $\mathcal{A}_s$ is also winning for her in the finitary Streett game on the same arena. The attractor of this region is also winning for her in this game.*
β : If Adam wins everywhere in the request-response game on an arena $\mathcal{A}_s$, then he wins everywhere in the finitary Streett game on the same arena.

Proof. α : In the winning region of Eve in the request-response game, she can use the strategy derived from [WHT03]. It guarantees that each request is matched by a corresponding response in the next $k \cdot n$ moves, and thus that the delay sequence is bounded. Thus $Win_E^{rr}(\mathcal{A}_s)) \subseteq Win_E^{fs}(\mathcal{A}_p)$. As the finitary condition is prefix-independent, we get $Attr_E(Win_E^{rr}(\mathcal{A}_s)) \subseteq Win_E^{fs}(\mathcal{A}_p)$. □

β : The construction of a winning strategy for Adam for finitary Streett from a strategy winning everywhere for him in request-response is similar to the one used to build a winning strategy for him in finitary parity. If π is a winning strategy for Adam in the request-response game, the strategy π' is defined by:

[3] At least, none that we were able to find.

1. Set b to 1.
2. Play the strategy π with initial memory from the state where the token is now until there is a sequence with a request followed by b moves without seeing the corresponding response.
3. Increment b.
4. Go back to step 2.

Once again a play that does not get stuck in step 2 is clearly winning for Adam. And a play that would get stuck in the step 2 would be a play consistent with π where each request is followed by a response, in contradiction with the fact that π is winning in request-response. Thus π' is winning for Adam everywhere in $\mathcal{A}_s$. □

From this lemma we derive Algorithm 3.

Algorithm 3. Algorithm computing the winning regions of Adam and Eve for the finitary Streett game

Require: Algorithm computing the winning regions in a request-response game
input $\mathcal{A}, p$
$\mathcal{B} \leftarrow \mathcal{A}$
repeat
$\mathcal{B} \leftarrow \mathcal{B} \setminus Attr_E(Win_E^{rr}(\mathcal{B}, p))$
until $Win_E^{rr}(\mathcal{B}, p) = \emptyset$
return $\mathcal{B}, \mathcal{A} \setminus \mathcal{B}$

As in the other algorithms, the number of times the loop is repeated is bounded by the number of states in the arena. The complexity is thus n times the complexity of the algorithm for request-response games, or $O(4^k \cdot k^2 \cdot m^2 \cdot n)$. This complete the proof of the following theorem:

Theorem 8. *Computing the winning regions in a finitary Streett game can be done in time* $O(4^k \cdot k^2 \cdot m^2 \cdot n)$.

In comparison, the reduction to a finitary parity game showed in [CH06] was running in time $O((n \cdot k! \cdot k^2)^{2k-3} \cdot m \cdot k! \cdot k^3)$. This reduction was based on the indexes of appearance records from [BLV96]. It also implies that the winning strategy of Eve in her region could use up to $k! \cdot k^2$ memory states. This makes the following corollary to our algorithm interesting:

Corollary 9. *There are winning strategies for Eve in her winning region that use no more than* $k \cdot 2^k$ *memory states.*

Proof. The strategy for Eve that derives from our algorithm is a combination of request-response strategies and attractors strategies. The request-response strategies use at most $k \cdot 2^k$, while the attractors strategies are memoryless. Furthermore, these strategies are combined in a *spatial* fashion: when the token goes from a region to another, the memory can be reseted. Thus Eve needs only as much memory as she needs to win the request-response games, *i.e.* $k \cdot 2^k$. □

Interestingly, the weak-Streett strategies for Eve need less memory, with 2^k memory states [NSW02], while classical Streett games need $k!$ memory states [DJW97, Hor05].

5 Conclusion and Developments

We gave algorithms that solve finitary parity and Streett games. They are much faster than their counterpart in the original paper by Chatterjee and Henzinger. The finitary parity problem, in particular, was proved to be in P, improving the former result of NP ∩ co-NP. The algorithm for Streett games represents a good improvement in time complexity, and yields more compact strategies for Eve. We had hoped to solve finitary Streett games with a Turing-reduction starting from weak-Streett games, which may have made the solution a PSPACE problem. However, if there is such a reduction, it eluded us so far.

Our next interests in this field of research are an extension of the notion of finitary games to Muller conditions, and the study of links between these games and a fragment of the ωBS-regular logic of Bojanczyk and Colcombet.

Acknowledgments. I wish to thank Olivier Serre for introducing me to finitary games and then helping me in the construction of the algorithms. Also, special thanks to Claire David, whithout whom I would never have met the deadline.

References

[AH94] R. Alur and T.A. Henzinger. Finitary Fairness In proceedings of *Logic In Computer Science*, LICS'94, p. 52–61. IEEE Computer Society, 1994.

[AHK02] R. Alur, T.A. Henzinger and O. Kupferman. Alternating-time temporal logic. In *Journal of the ACM*, volume 49, p.672–713. 2002.

[BC06] M. Bojanczyk and T. Colcombet. Bounds in ω-regularity In proceedings of *Logic In Computer Science*, LICS'06, p. 285–296, IEEE Computer Society, 2006.

[BLV96] N. Buhrke, H. Lescow and J. Vöge. Strategy Construction in Infinite Games with Streett and Rabin Chain Winning Conditions. In proceedings of *Tools and Algorithms for Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, TACAS'96, p. 207–224, Springer, 1996.

[CH06] K. Chatterjee and T.A. Henzinger. Finitary Winning in omega-Regular Games. In proceedings of *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *Lecture Notes in Computer Science*, TACAS'06, p. 257–271, Springer, 2006.

[Cha06] K. Chatterjee. Linear Time Algorithm for Weak Parity Games Technical Report No. UCB/EECS-2006-153. University of California at Berkeley, 2006.

[DJW97] S. Dziembowski, M. Jurdziński and I. Walukiewicz. How Much Memory Is Needed to Win Infinite Games ? In proceedings of *Logic In Computer Science*, LICS'97, p. 99–110, IEEE Computer Society, 1997.

[Jur00] M. Jurdziński Small Progress Measures for Solving Parity Games. In proceedings of *Symposium on Theoretical Aspects of Computer Science*, STACS'00, volume 1770 of *Lecture Notes in Computer Science*, p. 290–301, Springer, 2000

[Hor05] F. Horn. Streett Games on Finite Graphs. *Games in Design and Verification*, Workshop collocated with *Computer Aided Verfication*, 2005

[LT00] C. Löding and W. Thomas. Alternating Automata and Logics over Infinite Words. In proceedings of the *IFIP International Conference on Theoretical Computer Science*, IFIP TCS'00, volume 1872 of *Lecture Notes in Computer Science*, p. 521–535. Springer, 2000.

[MP92] Z. Manna and A. Pnueli. The Temporal Logic of Concurrent and Reactive System Springer, 2002.

[NSW02] J. Neumann, A. Szepietowski and I. Walukiewicz. Complexity of weak acceptance conditions in tree automata. In *Information Processing Letters*, volume 84, p181–187, Elsevier, 2002.

[Tho95] W. Thomas. On the Synthesis of Strategies in Infinite Games. In proceedings of *Symposium on Theoretical Aspects of Computer Science*, STACS'95, volume 900 of *Lecture Notes in Computer Science*, p. 1–13, Springer, 1995.

[VJ00] J. Vöge and M. Jurdziński. A Discrete Strategy Improvement Algorithm for Solving Parity Games. In proceedings of *Computer Aided Verfication*, CAV'00, volume 1855 of *Lecture Notes in Computer Science*, p. 202–215, Springer, 2000.

[WHT03] N. Wallmeier, P. Hutten and W. Thomas. Symbolic Synthesis of Finite-State Controllers for Request-Response Specifications. In proceedings of *Conference on Implementation and Application of Automata*, CIAA'03, volume 2759 of *Lecture Notes in Computer Science*, p. 11–22, Springer, 2003.

[Zie98] W. Zielonka. Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. In *Theoretical Computer Science*, volume 200(1-2), p. 135–183, 1998