Dicing on the Streett[⋆]

Florian Horn

LIAFA, Université Paris 7, Case 7014, 2 place Jussieu, F-75251 Paris 5, France
Lehrstuhl für Informatik VII, RWTH, Ahornstraße 55, 52056 Aachen, Germany
horn@liafa.jussieu.fr

Abstract. Streett/Rabin games are an adequate model of strong fairness in reactive systems. We show here some results about their stochastic version. We extend the known lower bound in memory for the pure winning strategies of the Streett player to randomized strategies. We also propose algorithms computing the almost sure winning regions of both players in stochastic Streett/Rabin games. The Rabin algorithm also yields directly a pure memoryless almost-sure winning strategy.

1 Introduction

Problems of control related to distributed resources often use the concept of *fairness*. It is underlying in classical frameworks such as mutual exclusion. In this situation, agents ask for access to a critical section, and then wait for their turn until the scheduler grants them access. This is *weak* fairness: a request needs to be fulfilled only if it is continuously enabled.

This notion of fairness is ill-adapted to online resources where both clients *and* providers are multiple: at any time, a client may choose another provider and drop his request. For example, a routing program confronted with an overloaded node will just change the path for its packages. Such abandoned requests need not to be addressed, yet a good controller must ensure that everyone gets a fair amount of the resource. The problem is then to distinguish, in finite time, a slow system from an unfair one. *Strong* fairness, or Streett condition, corresponds to this kind of specification: a system is strongly fair if each request that is repeatedly enabled is eventually fulfilled. Requests that are enabled only finitely often can be ignored. The complementary condition is called a Rabin condition. Thus, a Streett/Rabin game is played between an Streett player and a Rabin player.

Deciding if a state is winning for the Rabin player is a NP-complete problem [EJ88]. The NP membership comes from the existence of memoryless winning strategies for the Rabin player, and the NP-hardness is proven through a reduction from 3-SAT. By contrast, the Streett player may need memory to win. In [DJW97], the authors compute the exact memory requirements for all Muller games. In the case of Streett games, their result shows that winning strategies may need up to $k!$[1] memory states, on a game of exponential size. In [Hor05], we presented a family of games of polynomial size where winning strategies had to use at least $k!$ memory states.

Several algorithms were proposed for Streett games. The first one was a reduction to parity games via indices of appearance records [BLV96]. Later on, [Zie98] presented an algorithm for the more general Muller games, and [KV98] a reduction to the emptyness problem for weak alternating automata. In [Hor05], we presented a specialized version of the algorithm of [Zie98] for Streett games which can also be seen as an on-the-fly version of [KV98]. Finally, a rank algorithm inspired by the one in [Jur00] for parity games was presented in [PP06].

[⋆] Work supported by the EU-TMR network GAMES.

[1] In all the paper, k, n and m are the numbers of pairs, states and transitions in the game we consider.

This paper shows a factorial memory lower bound and a recursive algorithm for the qualitative solution of stochastic Streett games, thus extending the results of [Hor05] to the stochastic setting. In general, the lower bounds on memory of [DJW97] cannot be applied in the case of randomized strategies (see [CDH04]). And in the family of [Hor05], there is a randomized memoryless almost-sure winning strategy. In [CDH05], Chatterjee et al. showed that there are pure memoryless almost-sure winning strategies in stochastic Rabin games, while the Streett player needs no more than $k!$ memory. We present here a family of games where even randomized almost-sure winning strategies have to use at least $k!$ memory, providing a lower bound that matches the upper bound of [CDH05]. In the same paper, the authors also presented a reduction from stochastic Streett games to non stochastic ones, that allows to use the algorithms for non-stochastic games at small cost. The algorithm we present here extends the one of [Hor05] at no cost, and yields an alternative proof that pure memoryless strategies are enough for almost-sure winning strategies in stochastic Rabin games. However, using the algorithm of [PP06] in addition to the reduction of [CDH05] is still faster, albeit more expensive in space than our approach.

2 Definitions

An infinite $2\frac{1}{2}$-player[2] game G is a triplet $(V = V_A \cup V_E, E, Win)$ consisting of a finite graph (V, E) and a winning condition $Win \subseteq V^\omega$. A token is assumed to be in one of the states of V. It can only move along the edges. The set of states V is partitioned into Eve's states (V_E, denoted by circles), Adam's states (V_A, denoted by squares), and randomized states (V_R, denoted by diamonds). The owner of the state containing the token chooses the next state. If the state is randomized, the next state is chosen randomly[3] among its successors, with equal probabilities. 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, and we assume that every state has at least one successor. A play in $Win \subseteq V^\omega$ is winning for Eve. Otherwise, it is winning for Adam.

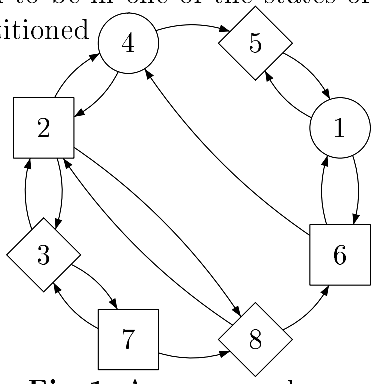


Fig. 1. A game graph

Definition 1. *A subgame of a game $G = (V, E, Win)$ is a game defined on a subset X of V such that each state in X has a successor in X, and each randomized state in X has all its successors in X. The edges and the winning set are restrictions of E and Win to X^2 and X^ω.*

A *basic Streett condition* is a pair of set of states $(Q, R) \subseteq V^2$. The set Q can be seen as a request, and the set R as the desired response. A Streett winning condition is a conjunction of basic conditions $(Q_i, R_i)_{i=1,\ldots,k}$. In such a game, Eve - the Streett player - wins if for each pair i, either the request set Q_i is visited only finitely often, or the response set R_i is visited infinitely often. Conversely, Adam - the Rabin player - wins if the token visits infinitely often one of the sets Q_i while the corresponding set R_i is visited only finitely often. A Streett game with k basic conditions is called a *Streett game of degree k.*

[2] The "half" player represents the randomized states.

[3] We decided here to consider only equal probabilities for each successor. This hypothesis could be loosened a lot, if one would want to investigate it. All our results depend only on the validity of property 3.

In this section, we introduce several notions about stochastic games. See [Tho95,Zie98] for more details on non-stochastic games, and [DA97,CJH03] for stochastic ones.

A *randomized strategy* for a player P is a function σ from V^*V_P to $2^V \setminus \{\emptyset\}$ respecting the edge relation: for all $q' \in \sigma(wq)$ we have $(q,q') \in E$. Informally, if P uses the strategy σ, then whenever he has to play after a prefix w, he chooses randomly[4] a successor in $\sigma(w)$. A play $\rho = q_1, q_2, \ldots$ is consistent with a strategy σ for P if for all i with $q_i \in V_P$ we have $q_{i+1} \in \sigma(q_1 \cdots q_i)$. A strategy is called *pure* (non-randomized) if $|\sigma(w)| = 1$ for all $w \in V^*V_P$. It is memoryless if the next move depends only on the current position of the token, that is, for all w, w' and q, $\sigma(wq) = \sigma(w'q)$. A strategy uses a memory of size m if it can be realized by a control automaton with m states: at each step, the next game state and memory state depend only on their previous values. A strategy σ for player P is *almost surely winning* if for any counter-strategy π of the other player, the probability that a play consistent with both σ and π belongs to Win_P is one. In the rest of the paper, whenever we use the qualifier *winning*, we mean *almost surely winning*. The winning region of a player P in a game G, denoted by $W_P(G)$, is the set of states from where he has a winning strategy. We do not consider here the problem of *optimal* strategies, which would involve computing probabilities of winning that are neither zero nor one: our only concern here is about almost-sure winning.

For a game G and a finite-memory strategy σ for player P, we call G restricted to σ for player P the $1\frac{1}{2}$-player game defined naturally by replacing the states belonging to P by randomized states whose possible successors are the ones prescribed by σ. In addition, the memory of σ is simulated by having multiple copies of each state, one for each memory value. Clearly, the projected paths of the restricted game are paths in the original game.

Definition 2. *An end-component K of a game graph is a strongly connected set of states such that each successor of a randomized state in K also belongs to K.*

Property 3. [DA97] For any strategies σ, π of Eve and Adam, the probability over the plays consistent with σ and π that the set of states visited infinitely often is an end-component is one.

An end component is consistent with a finite memory strategy σ for player P if it is an end-component of G restricted to σ for player P. In Muller games, where the winner depends only on the set of states visited infinitely often, we define the winner of an end component K: it is the winner of a play that visits infinitely often each state in K, and only these. In the special case of Streett/Rabin games, an end component is winning for Eve if for each appearing request, a matching response appears. Otherwise, it is winning for Adam.

For strategies with finite memory on Muller games, an interesting (and non-trivial) corollary to Property 3 is that a strategy is winning if and only if each consistent end-component is winning [DA97].

The *attractor of W for player P in the game G*, denoted $Attr_P^G(W)$, is the set of states where P can ensure that the token has a positive probability to reach the set W. It is computed inductively as usual: $W_0 = W$ and W_{n+1} is the union of W_n, $\{q \in V_P \cup V_R \mid \exists q' \in W_n, (q,q') \in E\}$ and $\{q \in V \setminus (V_P \cup V_R) \mid \forall q' \in W_n, (q,q') \in E\}$. The *attractor strategy* for player P consists in always going from a state of W_n to a state in W_{n-1}, thus getting closer to W. It is pure and memoryless. The following property is a direct consequence of the definition of attractors:

[4] More complex strategies could be considered, but they are not needed for almost-sure winning.

Property 4. An end-component consistent with the attractor strategy to W for player P contains either a state of W or no state of $Attr_P(W)$.

A *trap* is a set from which one of the players cannot escape: A set $T \subseteq V$ is a trap for player P if each state of T belonging to the other player has a successor in T, and each other state (in V_P or V_R) has *all* its successors in T. Notice that the complement of an attractor is a trap for the same player, and that a trap is always a subgame.

Remark 5. While the attractor is a *weak* notion (the token may escape the attractor without visiting the desired target set, even if the attractor strategy is used), the trap is a *strong* one (the token will never leave a trap if the untrapped player does not allow it to do so).

3 A memory lower bound for randomized strategies in Streett games

In [DJW97], the authors address the problem of memory in the more general case of Muller games. From their results, one can derive a family of Streett games where the k^{th} representative is of degree k and any winning strategy for Eve uses at least $k!$ memory states. This game has $O(k^2 \cdot k!)$ states. In [Hor05], we present a family of quadratic size with the same properties.

In [CDH04], the authors show that the lower bound on memory from [DJW97] does not hold for randomized strategies. They also show that 1-player and $1\frac{1}{2}$-player Streett games admit memoryless randomized winning strategies, while pure winning strategies need a memory of size k. It is natural to ask whether something similar happens with 2-player Streett games. Note that the family of [Hor05] does not answer this question, as it admits memoryless randomized winning strategies.

In this section, we present a family of 2-player games, where the k^{th} representative has degree k (and $O(k^2)$ states) and any winning randomized strategy for Eve uses at least $k!$ memory. This matches the upper bound of [CDH05], which showed that Eve had winning strategies with $k!$ memory states, even in the case of $2\frac{1}{2}$-player games.

Theorem 6. *For any k, there is a 2-player Streett game G of degree k and size $O(k^2)$ such that any winning randomized strategy for Eve uses at least $k!$ different memory states*[5].

3.1 Presentation of the family

The game graph G_k is presented in Figure 2. Note that we do not use randomized states. It is a succession of basic loops:

- In the initial state $\aleph$, Adam chooses two basic Streett conditions i and j, where $i \neq j$.
- Eve must add a request to either i (by going to i_j) or j (by going to j_i).
- Adam must answer to either i (by going to the state i) or j (by going to the state j).

For each G_k, there is a winning strategy σ_k for Eve that uses $k!$ memory states:

- The memory state is identified with a permutation $\pi \in \mathcal{S}_k$
- In the state $\{i, j\}$, Eve moves to the state i_j if i appears first in π, and to j_i otherwise.
- In the state i, Eve updates her memory by moving i at the beginning of π.

To show that σ_k is winning, note that each time a condition i is requested, and does not immediately get a response, it will get farther in π. As it will not go back to the beginning unless it gets a response, we can conclude that there can be at most k successive requests Q_i without a response R_i. Thus σ_k is winning.

[5] Actually, Adam can win with probability 1 against any strategy with less than $k!$ memory.

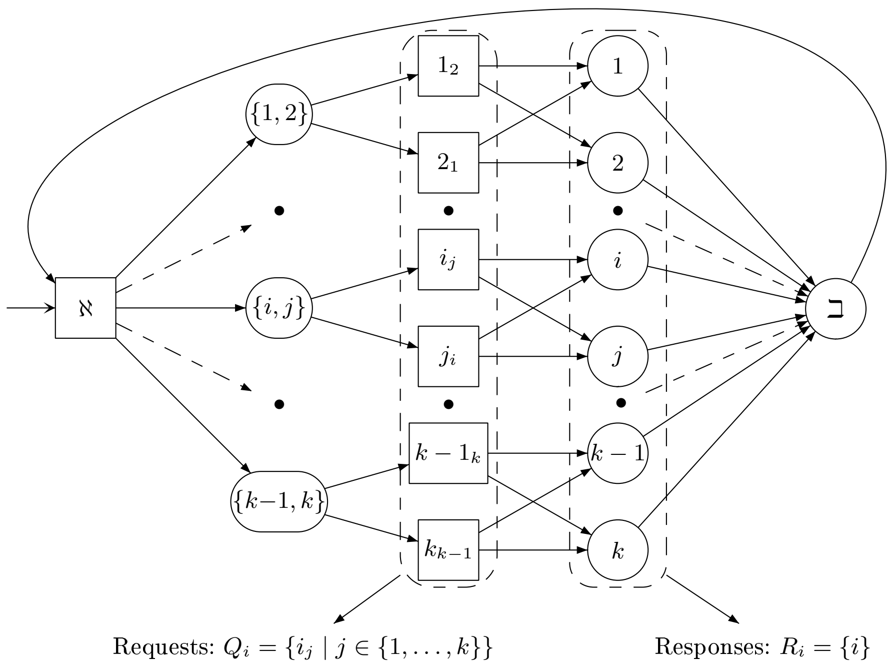


Fig. 2. Game G_k

3.2 Proof of the lower bound

Proposition 7. *Any winning strategy for Eve in G_k needs at least $k!$ different memory states.*

Let us fix for this section a winning finite-memory strategy σ for Eve, possibly randomized. The point now is to exhibit strategies for Adam against which σ must behave in different ways. For this we use the notion of high priority condition.

Definition 8. *A condition i is a* high priority condition *(HPC for short) for σ in the memory state p if in every finite play ρ that 1) is consistent with σ, 2) starts in state $\aleph$ with memory state p and 3) ends with the request i, there is at least one occurrence of a response i.*

Intuitively, the above definition states that if i is an HPC, then regardless of what happens in Adam's states and random states, the token will not visit Q_i before it had visited R_i. Notice that there cannot be two different HPC i and j in the same memory state, as Eve has to move to either i_j or j_i if Adam goes from $\aleph$ to the state $\{i,j\}$.

Proposition 9. *For each condition i, there is a memory state p such that i is an HPC for σ.*

Proof. Consider G_k^σ, the game G_k restricted to σ. It is a $1\frac{1}{2}$-player game, whose states are of the form (q,p) where q is a state of G_k and p a memory state of σ_k. The property *The condition i is not an HPC for σ in the memory state p* is equivalent to *There is a path in G_k^σ from $(\aleph,p)$ to a state in Q_i that does not go through a state in R_i.*

Assume now by contradiction that, for each memory state p at the initial configuration $\aleph$, the condition i is not an HPC. Consider G_k^σ restricted to Adam's strategy "Avoid R_i and otherwise, act randomly". It is a Markov chain where there is no edge leading to a state of R_i. On the other hand, there is still a path to a state of Q_i from every $(\aleph,p)$. As the end-components of a Markov chain are the bottom strongly connected components (scc's without successors outside of itself), it follows that each of these includes a state of Q_i, and no state of R_i. Thus Adam wins with probability one, which contradicts the assumption that σ is winning. □

By Proposition 9, we know that there is a memory state in which i is an HPC. Suppose that, once Eve goes to this memory state, Adam chooses never to go again to a state $\{i, j\}$. The token cannot reach the state i, so i will remain an HPC. Meanwhile, Adam and Eve play in a subgame that is isomorphic to G_{k-1}, so Eve needs memory $(k-1)!$ for winning. As a memory state cannot have two different HPC's, Eve needs at least a total memory of size $k!$.

4 Algorithms

In this section, we describe an algorithm which computes the winning region of Eve in $2\frac{1}{2}$-player games. It is an adaptation of the algorithm for 2-player games presented in [Hor05]. The solution for these games can also be achieved by using the reduction from stochastic Streett games to non-stochastic ones presented in [CDH05], along with any algorithm solving non-stochastic Streett games ([BLV96], [Hor05], or [PP06]). The cost of the reduction is small (from n states, m transitions and k basic conditions to nk states, mk transitions and $k+1$ basic conditions), but we show here that our algorithm of [Hor05] can be extended to stochastic games at no extra cost at all.

4.1 An algorithm for Streett games

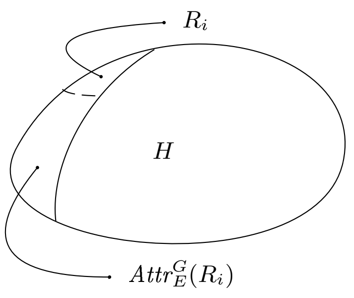
The main idea of our algorithm is that Adam cannot win if Eve can reach each of the R_i from anywhere with a positive probability. Let's note already that Adam *may* win while being in all the R_i attractors, if one of these attractors does not cover the whole graph. Otherwise, Eve can win by playing the attractor strategy to R_1, then, once a state of R_1 has been reached, switching to the attractor strategy to R_2, and so on, restarting once each R_i has been visited. The only end-components that are consistent with this strategy intersect each of the R_i (see Property 4). Thus Eve would win with probability one.

From this, we conclude that if Adam's winning region is non-empty, it must contain a state outside of one of the R_i-attractors. Thus we can first ask that he chooses a condition i and plays outside of the R_i-attractor. This defines a subgame, that we call H, which is a trap for Eve. In this subgame, Adam can win either by visiting infinitely often Q_i or by winning with respect to the other conditions. Similarily, Eve may have a non-empty winning region in H only if she has a winning state outside the H-attractor of Adam to Q_i. This defines a second subgame K which is a H-trap for Adam, and has no occurrence of Q_i. We can thus compute the winning region of Eve in K using the algorithm recursively. As K is a trap for Adam in H, the winning region of Eve in K is included in her winning region in H. We now remove it and its attractor, and repeat this procedure to obtain a decreasing succession of traps for Eve, removing at each step a non-empty set of states winning for her in H. This decreasing sequence can end either with the empty set or not. In the first case, we have to check what happens with the other conditions. In the second case, we claim that the final set is in the winning region of Adam. Finally, if it ends in an empty set for each condition, we claim that Eve wins everywhere. The full procedure is formally written in Algorithm 1. For the sake of clarity, Figure 3 shows more visually the workings of the internal loop.

Algorithm 1 Algorithm computing the winning region of the Streett Player

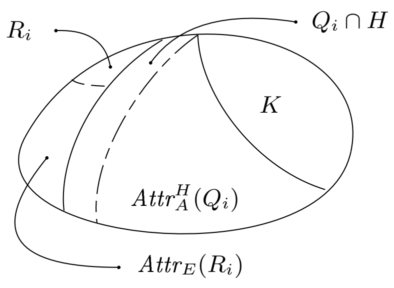
```
Require: Algorithms to compute attractors and remove sets of states
  input $G, Win$
  for all $i \in \{1, 2, .., k\}$ do
    $H \leftarrow G \setminus Attr_E^G(R_i)$
    repeat
      $K \leftarrow H \setminus Attr_A^H(Q_i)$
      $L \leftarrow WinStreett(K, Win \setminus \{Q_i, R_i\})$
      $H \leftarrow H \setminus Attr_E^H(L)$
    until $L = \emptyset$
    if $H \neq \emptyset$ then
      return $WinStreett(G \setminus Attr_A^G(H), Win)$
    end if
  end for
  return $G$
```

1. The subgame H has been set to $G \setminus Attr_E^G(R_i)$ in the main loop. It is a trap for Eve.



2. The subgame K is set to $H \setminus Attr_A^H(Q_i)$. It is a trap for Adam in H, but not in G.

3. The pair (Q_i, R_i) does not appear in K. L is set to the winning region of Eve in K, computed recursively.



4. If L is not empty, $Attr_E^H(L)$ is removed from H for the next iteration of the **repeat** loop (step 2).

5. If both L and H are empty, the algorithm proceeds to the next iteration of the **for all** loop.

6. If L is empty, but H is not empty, the **for all** loop is broken, and the algorithm is called recursively on $G \setminus Attr_A^G(H)$.

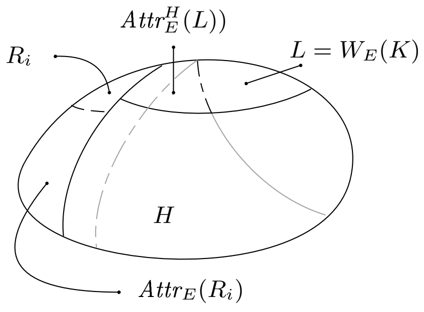
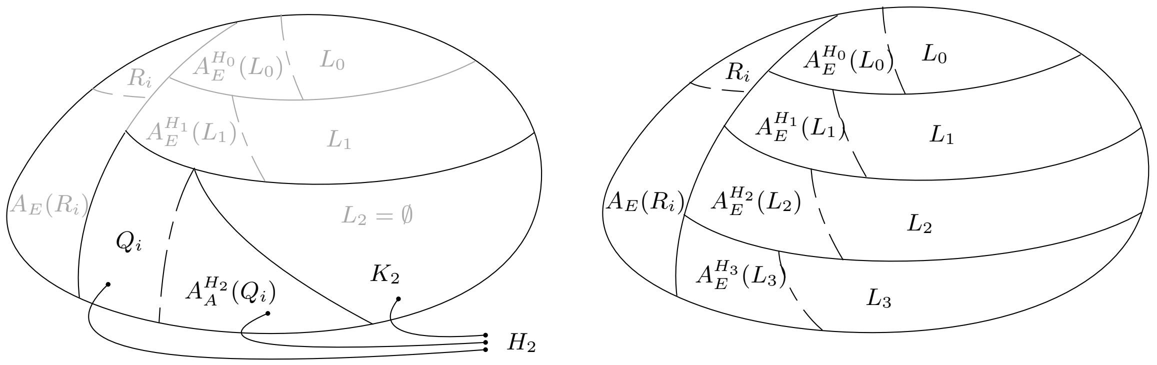


Fig. 3. First execution of the **repeat** loop for the condition i

4.2 Correctness

The validity of Algorithm 1 depends on the lemmas 10 and 11 below, which state properties about G when the internal loop ends. A representation of the two ending cases (depending whether H is empty or not) is given in Figure 4. For better readability, we indexed the names of the sets with the number of the iteration of the loop in which they were defined, and replaced the symbol *Attr* by just A.



(a) H_2 is winning for Adam (b) Reaching an empty region

Fig. 4. End of a run of the internal loop

Lemma 10. *When the internal loop of Algorithm 1 ends, H is winning for Adam.*

Proof. If H is empty, the result is trivial. We can thus restrict ourselves to the case where H is *not* empty, and G looks like Figure 4(a).

Adam uses the attractor strategy in $Attr_A(Q_i)$ and his winning strategy in the subgame K. Note that Adam wins with probability 1 in K, since $L = \emptyset$, (see Theorem 8 in [DAH00]). Moreover, H is a trap for Eve, and Adam's strategy does not move out of it. Thus, the end-components consistent with it are included in H. The cases to consider now concern the presence of a state of $Attr_A(Q_i)$ in the end-component: if there is one, Property 4 states that there must also be a state of Q_i in the end-component; if there is none, the end-component is an end-component of K, consistent with Adam's winning strategy for this subgame.

In both cases Adam wins with probability 1. □

Lemma 11. *If H is empty at the end of each internal loop of Algorithm 1, Eve wins everywhere.*

Proof. If H is empty at the end of the internal loop for condition i, G looks like in Figure 4(b). We suppose here that this is the case for each i.

We now describe a winning strategy σ_E for Eve in G. It uses a top-level memory of size k to switch between k substrategies $\sigma_1 \ldots \sigma_k$, each of these using a memory of size $(k-1)!$. The strategy σ_i is described below:

- In the subgames L: the subgame strategy (memory $(k-1)!$).

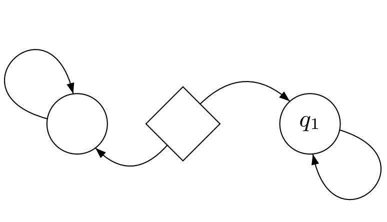
– In the attractors $Attr_E^H(L)$: the attractor strategy (memoryless).
– In $Attr_E(R_i)$: the attractor strategy (memoryless).
– In R_i: Eve updates her top-level memory to $(i+1)$ *mod* k and applies $\sigma_{i+1 \ mod \ k}$.

There are two cases: Either the token goes infinitely often in each of the R_i, or Eve's top-level memory is ultimately constant (at a value i) after a finite prefix. In the first case, she wins. In the second case, we consider the end-components consistent with the strategy σ_i.

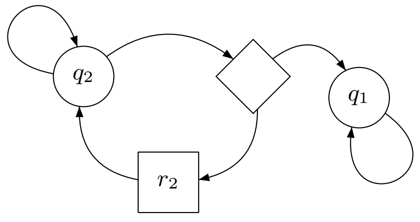
We claim that each of them is included in one of the subgames L. Note that each L is a trap for Adam in the current subgame H, and the token can escape from L only by going to an *earlier* H (defined earlier in the internal loop). Consider now the first H of the internal loop that intersects the end-component. Property 4 says that the end-component intersects the associated L. If the end-component is not included in this L, then by connexity it should intersect an earlier H, which contradicts the minimality of H. Thus, Eve wins with the strategy of the subgame L. □

4.3 Algorithm for the Streett fighting man

We want now to compute the almost-sure winning region of Adam. In the course of algorithm 1, we found some regions where Adam was winning with probability 1 (see lemma 10). However, it is not true that the complement of Eve almost-sure winning region is winning for Adam (see figure 5(a)). The reason is that the winning region we find are expanded through a *weak* attractor, that corrupts the remainder of the computation. A natural idea would be to replace it with a **strong** attractor[6], where the target set is reached with probability 1. This doesn't work either, as illustrates figure 5(b): If a variant of Algorithm 1 with **strong** attractors starts by considering condition 1, it will remove only the rightmost state as winning for Adam. The remainder of the graph is now winning for Eve, though Adam can also win with probability 1 in this region.



(a) Equal chances to win



(b) Inadequacy of the **strong** attractor

Fig. 5. Motivation for Algorithm 2.

Our solution is to add an external loop to algorithm 1:

Algorithm 2 Algorithm computing the winning region of the Rabin player

```
Require: Algorithms to compute attractors and remove sets of states, Algorithm 1
  input $G, Win$
  while $WinStreett(G) \neq \emptyset$ do
    $G \leftarrow (G \setminus Attr_E(WinStreett(G)))$
  end while
  return $G$
```

[6] The computation of such an attractor can be done through the computations of $2n$ *weak* attractors

This algorithm works by removing almost-sure winning regions for Eve and the corresponding attractors. None of these regions is a likely candidate to an almost-sure winning region of Adam. When it cannot find a winning region for Eve, we claim that the remainder of the graph is winning for Adam. Indeed, in this case, the computation of Algorithm 1 yields a game graph only composed of a succession of region winning for Adam and the corresponding attractors (much like Figure 4(b) without the leftmost region). By the argument we used for Figure 4(b), the token will eventually end in one of the subgames, and Adam will win with probability one.

Notice that the strategy of Adam in the attractor is memoryless, as is the one in the subgames (described in the proof of lemma 10). This yields an alternative proof, without a reduction to a 2-player game, for the following theorem of [CDH05]:

Theorem 12. *For any stochastic Streett/Rabin Game, if Adam has an almost-sure winning strategy, then he also has an almost-sure winning strategy that is pure and memoryless.*

4.4 Time and space complexity

Algorithm 1 may have to call k different runs of the internal loop to remove a single state in G, if it guesses wrong. With the same worst-case scenario, the internal loop has to call an attractor and a recursive call to the algorithm to remove one state in G_i. The attractor computation being linear in the number of transitions m, we get the equation $C(n,k) = k \cdot n \cdot (O(m) + n \cdot C(n-1, k-1))$ for a loop of the main algorithm. Thus $C(n,k) = O(m \cdot n^{2k} \cdot k!)$, and the time complexity needed to compute all the states is $O(m \cdot n^{2k} \cdot k!)$. For the winning region of the Rabin player, we need one more loop, yielding a total complexity of $O(m \cdot n^{2k+1} \cdot k!)$. Our algorithm uses memory in the form of variables. There are four variables in our version, and each holds a subgame, which can be represented as a set of states (size $O(n)$). As there are at most k nested versions of the algorithm running simultaneously, the space needed by our algorithm is $O(n \cdot k)$. One can improve this space easily by noticing that each time a subgame is defined, it is smaller than the subgames defined before. Thus we need only to remember when a state was removed. This reduces the space complexity to $O(n \cdot \log k)$. Notice that this allows only to compute the winning regions. Recording the strategy takes $O(n \cdot k! \cdot \log n)$. It can be done in the same time, though.

The three algorithms we mentioned for Streett games ([BLV96], [Hor05] and [PP06]) each have an advantage. The reduction to a parity game doesn't have a good upper bound in time, and uses a lot of memory. Yet, it could still be the fastest if the parity problem were in P. The best upper bound in time is the algorithm of [PP06], with $O(n^{k+1}k!)$. However it also uses a lot of space: $O(n \cdot k \cdot k! \cdot \log(n))$. Our algorithm achieves a balance between time and space complexities. These three algorithms can be used along with the reduction of [CDH05], which transforms a stochastic game of size (n, m, k) to a non-stochastic one of size $(nk, mk, k+1)$. This paper allows us to circumvent the reduction when we use the algorithm of [Hor05], reducing the complexity from $km \cdot (kn)^{2k+2} \cdot (k+1)!$ to $m \cdot n^{2k} \cdot k!$. However, it is not enough to change the fact that even with the cost of the reduction, [PP06] is still faster, and the reduction could still be even faster, should the parity games problem be in P.

5 Conclusion

We have extended several results from the non-stochastic Streett/Rabin games to the stochastic setting: factorial lower bound on the memory for Streett games strategies, and a direct

algorithm for determining the winning regions and strategies. We intend to extend these results to infinite game graphs (*e.g.* pushdowns graphs), and to extend them for Muller games.
Acknowledgments. I wish to thank Wolfgang Thomas and Anca Muscholl, for their support in both research and writing. I would also like to thank the referees, whose comments helped me a lot in (re)writing, as well as Julien Cristau and Olivier Serre, who managed to find the `tex` source that I managed to lose.

References

[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.

[CDH04] K. Chatterjee, L. de Alfaro and T.A. Henzinger. Trading Memory for Randomness. In proceedings of *Quantitative Evaluation of Systems*, QEST'04, p. 206–217, IEEE Computer Society, 2004.

[CDH05] K. Chatterjee, L. de Alfaro and T.A. Henzinger. The Complexity of Stochastic Rabin and Streett Games. In proceedings of *International Conference on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, ICALP'05, p. 878–890, Springer, 2005.

[CJH03] K. Chatterjee, M. Jurdziński and T.A. Henzinger. Simple Stochastic Parity Games. In proceedings of *Computer Science Logic*, volume 2803 of *Lecture Notes in Computer Science*, CSL'03 p. 100–113, Springer, 2003.

[DA97] L. de Alfaro. Formal Verification of Probabilistic Systems. Ph.D. Dissertation, Stanford University, 1997.

[DAH00] L. de Alfaro and T.A. Henzinger. Concurrent Omega-Regular Games. In proceedings of *Logic In Computer Science*, LICS'00, p. 141–154, IEEE Computer Society, 2000.

[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.

[EJ88] E. A. Emerson and C. S. Jutla. The Complexity of Tree Automata and Logics of Programs. In proceedings of *Foundation Of Computer Science*, FOCS'88, p. 328–337, IEEE Computer Society, 1988

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

[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

[KV98] O. Kupferman and M. Y. Vardi Weak Alternating Automata and Tree Automata Emptiness. In proceedings of *Symposium on the Theory of Computing*, STOC'98, p.224–233, Association for Computing Machinery, 1998.

[MS95] D. E. Muller and P. E. Schupp. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra. In *Theoretical Computer Science*, volume 141(1-2), p. 69–107, 1995.

[PP06] N. Piterman and A. Pnueli. Faster Solutions of Rabin and Streett Games. In proceedings of *Logic in Computer Science*, LICS'06, IEEE Computer Society, 2006.

[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.

[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