Streett Games on Finite Graphs*

Florian Horn

LIAFA, Université Paris 7, France & LI7, RWTH Aachen, Germany

Abstract. Streett Games are an adequate model of strong fairness in reactive systems. We show that solving these games is co-NP complete, and that they require memory factorial in the size of the winning condition, even when the size of the game is polynomial. Two algorithms, computing the strategies of both players, are also presented.

1 Introduction

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

These problems can also be specified with another kind of fairness, called Request-Response fairness (RR for short) which is presented in [WHT03]. In this case, each request that is enabled once must eventually be fulfilled. RR-fairness allows more compact specifications (translation to weak fairness is exponential), but strategies for RR-games need exponential memory.

These notions of fairness are ill-adapted to online resources where both clients *and* providers are multiple: at any time, a client can choose another provider and drop his request. For example, a routing program, confronted with an overloaded node, will just change the path of its packages. Such abandoned requests need not to be addressed, yet a "good" controller must ensure that everyone gets a fair amount of the ressource. The problem is then to differentiate a slow system from an unfair one. The notion of *strong* fairness, or Streett condition, corresponds to this kind of specification: a system is strongly fair if a request that is repeatedly enabled must eventually be fulfilled. Requests that are enabled only a finite number of times can be ignored without penalty.

Of course, infinite paths are of no practical use: the point is to decide in finite time if a controller is strongly fair, i.e. whether a faithful client can be sure that his request will eventually be fulfilled. The question of strong fairness in closed systems has been studied in [Saf92] and [Sch02]. In [Zie98], the author addresses the problem of the more general Muller games on infinite graph. As for strategies, it is known that Muller games and in particular Streett games require memory, whereas parity games and Rabin games (Streett games for the other player) do not. In [DJW97], the amount of memory needed for any given Muller condition

* Work supported by the EU-TMR network GAMES

is considered. A solution for Streett games through a reduction to parity games is given in [BLV96]. This reduction uses the concept of Index of Appearance Records presented in [MS95]. It leads to a factorial blow-up, with a parity game of size $n.k^2.k!$, with $2k$ colors, for n states and a condition of size k. The solution of parity games is equivalent to model-checking for μ-calculus and was extensively studied. We will compare our results with the behaviour of the IAR reduction using two main algorithms. One, unsing "Strategy Improvement" is presented in [VJ00] has the best results with a polynomial complexity in practical cases, but without theoretical proof of this behaviour. The other one, presented in [] has the best proven complexity with $O(n^{d/2})$, where d is the number of colors in the game. The complexity for the solution of Streett games is then $(n.k^2.k!)^k$, while with the strategy improvement, the (unproven) complexity is a polynom in $n.k^2.k!$. Finally, in [EJ88], a result about Rabin tree automata translates as co-NP completeness for Streett games.

This paper presents two new contributions: first we show a family of games where winning strategies require memory factorial in the size of the Streett condition. The existence of such a family can already be established using [DJW97], but our example does not entail an exponential blow-up. Second, we present a better theoretical complexity for solving Streett games through a new algorithm, that appears to be a specialized version of the one for infinite Muller games in [Zie98]. Its worst case complexity is $n^k.k!$. This is better than the use of the IAR's reduction with a $O(n^{d/2})$ algorithm, but worse than the use of the Strategy Improvement if it is indeed polynomial.

2 Definitions

2.1 Games and strategies

An infinite two-player game G consists of a finite[1] graph (Q, E) containing a token. The token is always in one of the states and can only move along the edges. In this article, the two players will be called Eve and Adam. Q is partitioned into Eve's states (Q_E), represented as circles, and Adam's states (Q_A), represented as squares. The owner of the state containing the token chooses the next state. A play $\rho = q_1, q_2, \ldots$ is the infinite sequence of states visited by the token, thus respecting the edge relation: for all i, $(q_i, q_{i+1}) \in E$. We consider only infinite plays, and we assume that every state has at least one sucessor. Finally a game is defined by $Win_E(G) \subseteq Q^\omega$, the set of winning plays for Eve. All the other plays are winning for Adam.

An example of game graph is given in figure 1.

[1] Although many results are still valid on infinite graphs.

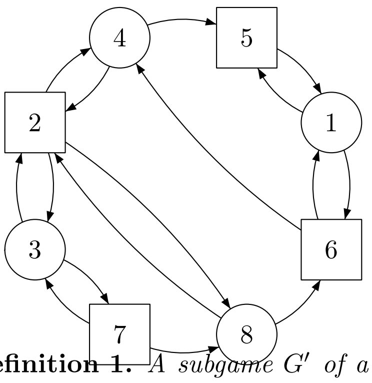


Fig. 1. A game graph

Definition 1. *A subgame G' of a game G is a game such that $Q' \subseteq Q$, $E' = E|_{Q' \times Q'}$ and $Win'_E = Win_E \cap Q'^{\omega}$.*

Remark 2. Our definition of a game imposes that every state has a successor, so it does not hold that every subgraph is a subgame.

Usual problems about games are to compute the winning regions and the corresponding strategies. In this section, we will describe several tools that will be useful in this article. See [Tho95] and [Zie98] for more details.

A strategy for a player P is a function σ from Q^*Q_P to Q respecting the edge relation: $(q, f(w.q)) \in E$. Informally, it decides which move P must take after any finite play ending in a state belonging to P. A play $\rho = q_1, q_2, \ldots$ is consistent with a strategy σ for P if $\forall i, q_i \in Q_P \Rightarrow q_{i+1} = \sigma(q_1 \ldots q_i)$. A strategy is *winning* if any play consistent with the strategy is in Win_P. It is *positional* (or *memoryless*) if the next move depend only on the current position of the token. A strategy uses a memory m if it can be represented by a control automaton with m states: at each step, the next move and the next control state depends only on the current position of the token and the current control state.

The winning region for Eve (resp. Adam) is the set of states from where she (resp. he) has a winning strategy. We write $W_E(G)$ (resp. $W_A(G)$) to denote this set.

A player often needs to reach or to avoid particular sets of states. We will call $Attr_P^G(V)$ (read "the attractor of V for player P in the game G") the set of states where P can force the token to reach the set V. The corresponding "attractor strategy" is positional.

Definition 3. *A trap $T \subseteq Q$ for a player P is a subgame that P cannot leave: $\forall q \in T \cap Q_P, \forall q' \in Q, (q, q') \in E \Rightarrow q' \in T$.*

Remark 4. The complement of an attractor for P is a trap for P.

Remark 5. Let T be a trap for a player (say Adam) in a game G, and σ be a strategy for Eve in that trap. Then the sets of plays consistent with σ in the game G and in the subgame T are equal. In particular, a winning region for Eve in the subgame T is still a winning region for her in G. Another crucial use for our purpose is that a trap for Adam relatively to another trap for Adam is still a trap in the general game.

2.2 Winning Conditions

Two kinds of winning conditions will be considered in this article. Both are *prefix independent*: for any prefix $w \in Q^*$ and any play $\rho \in Q^\omega$, ρ and $w\rho$ are winning for the same player. Specifically, strategies do not depend on the initial memory.

Parity: In a parity game a coloring function associates an integer to each of the states. To each play corresponds a set of colors appearing infinitely often. If the greatest color in this set is even, then Eve wins. If not, Adam does.

Streett: A *basic* Streett condition is a pair of set of states (Q, R). The set Q corresponds to requests, and R are the desired responses. A general Streett condition is a conjunction of basic Streett conditions $(Q_i, R_i)_{i=1\dots k}$. To win such a game, Adam must force the token to visit infinitely often one of the requests Q_i while the corresponding answer R_i is visited only a finite number of time. Conversely, Eve wins if for every request Q_i visited infinitely often, the corresponding answer R_i is visited infinitely often. Note that *finite* occurrences of requests or responses are not considered: Streett games translate the notion of *strong* fairness.

2.3 Index of appearance records (IAR)

The IAR reduction from [MS95], applied to games in [BLV96], is described below:

Definition 6. *An IAR for a Streett condition made of k basic conditions is a tuple (π, e, f) where π is a permutation over $\{1 \dots k\}$ ($\pi \in \mathcal{S}_k$) and e, f are two integers in $\{1 \dots k\}$*

IAR's are used to reduce a Streett game to a parity game with the following rules: the states of the parity game are $Q \times \mathcal{S}_k \times \{1 \dots k\}^2$. The color of a state (q, π, e, f) is $c = \max(2e, 2f - 1)$. To each transition (q, q') of the Streett game correspond all the transitions $((q, \pi, e, f), (q', \pi', e', f'))$ which respect:

- π' is obtained from π by shifting all the conditions fulfilled in q' (i.e. $q' \in R_i$) in the beginning of the permutation.
- e is the greatest position in π of a condition fulfilled in q' (i.e. $q' \in R_i$).
- f is the greatest position in π' of a condition requested in q' (i.e. $q' \in Q_i$).

The order π corresponds to the "age" of each condition, with the younger ones in the beginning, and the older ones in the end. The "age" considered here is the number of steps since the last time a condition was fulfilled. With this definition, the winner of a play in the parity game is the winner of the projection of this play on the original game. The size of the parity game is $n.k!.k^2$.

3 Optimality of the $k!$ memory

In this section, we describe a family of Streett games realizing the factorial upper bound of the memory needed by Eve. The existence of such a family could already be deduced from [DJW97] where the game graph has exponential size in the number of Streett pairs. Here we present a family of games where the size has only a quadratic growth-rate.

As in the IAR reduction, the point is to give priority to the "oldest" conditions, to ensure that each one is fulfilled often enough. Factorial memory naturally appears as an order between the conditions that guides Eve's decision.

3.1 A Family of Games Requiring Factorial Memory

We present a simple family of games $(G_k)_{k\in\mathbb{N}}$ expressing the heart of the problem: Adam asks for two conditions at the same time, and Eve can answer to only one of them. After that, the token goes back to the beginning, and so on. The corresponding graph is given in figure 2.

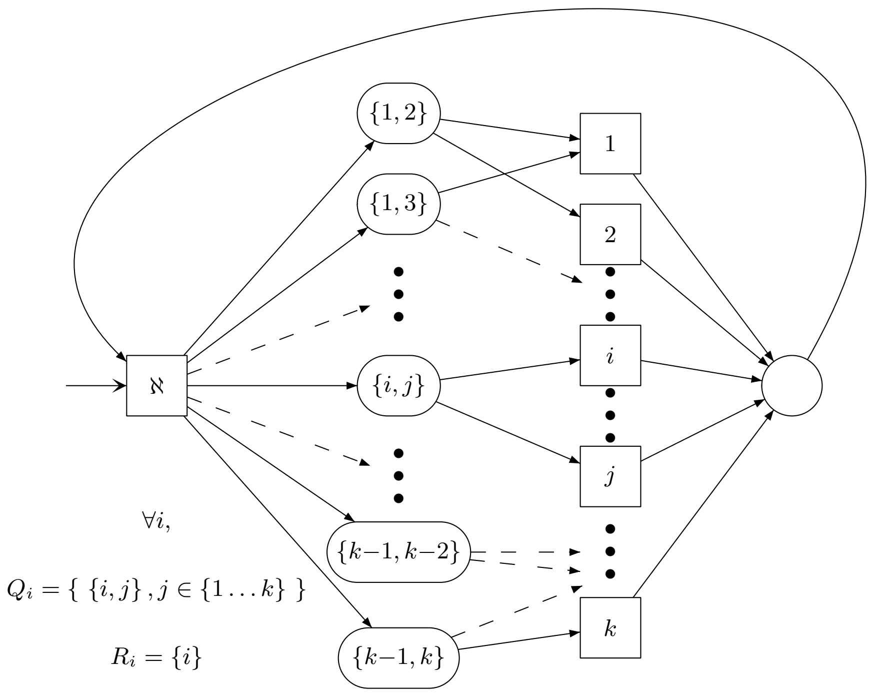


Fig. 2. Game G_k

These games can be solved by a strategy with $k!$ memory. The natural strategy σ_0 is defined as follows :

– The memory state is identified to a permutation $\pi \in \mathcal{S}_k$
– In the state $\{i, j\}$, Eve moves to the state i or j appearing first in π
– The memory is updated by shifting the chosen state at the end of π.

The permutation π behaves as a "priority queue": when Eve must choose between two conditions, she favors the oldest one, and changes π such that this condition now has the lowest priority.

The argument proving that σ_0 is winning is the following: each time a condition is asked and not immediately fulfilled, it gets strictly lower in the priority queue, and will get higher only when it is fulfilled. Thus no condition can be infinitely often requested without being also infinitely often fulfilled.

3.2 Proof of the lower bound

Memory strategy for Adam The main item of this proof is to create a strategy for Adam that forces Eve to go through the node $\aleph$ with $k!$ different contents of her memory. This strategy is built by iteration and for a given winning strategy σ. For the remainder of the proof we suppose that Eve plays accordingly to this strategy. The initial step is the game G_2 with two conditions: no positional strategy can be winning. The iteration step is the object of this section, and uses the concept of high-priority condition (HPC for short). Notice that, although we prove that the behaviour of σ has to be close to σ_0's one, we do *not* suppose anything about σ, except that it is winning.

Definition 7. *A condition i is an HPC in the memory state p if for any play ρ starting in p, the first occurence of any of the states $\{i, j\}$ in ρ is followed by i.*

By extension, we will say that i is an HPC after a finite play w if it is an HPC in the memory state of Eve at the end of w.

Remark 8. The strategy σ_0 has π_0 as an HPC in each memory state π.

Proposition 9. *Let σ be a winning strategy of Eve. The following hold:*

α : $\forall i$, Adam has a strategy τ_i such that as long as i is not *an HPC, then i will be requested and never be answered in the play defined by σ and τ_i.*
β : If i is an HPC after a finite play, then it will stay an HPC until it is requested (and fulfilled, by definition of an HPC).
γ : There can be only one HPC in a given memory state.

Proof. α : If i is not an HPC, then by definition, there exists a play ρconsistent with σ that leads to an unanswered request of i. Eve's moves are defined by σ. The strategy τ_i links each prefix of ρ ending in an Adam node to the next state[2].

[2] This supposes that Adam can choose his move knowing σ and the current state of Eve's memory, but it is not a problem since σ is winning against *any* moves of Adam.

β : Let u, v, w be finite plays such that $w = uv$. Suppose that i is an HPC after u and not after w. Then by definition, all plays starting by u have their first occurence of a $\{i, j\}$ followed by i, while this does not hold for w. Hence, in v, there is at least one occurence of a $\{i, j\}$ (the first one is followed by i).
γ : If i and j are both HPC's, there is a contradiction if Adam requests $\{i, j\}$.

The main argument of the proof is that Adam can always play in order to transform any condition i into an HPC. He only needs to play according to the strategy τ_i defined in Proposition 9-α, which leads to a state where i is requested and not fulfilled. He can do so repeatedly as long as i is not an HPC. If i never became an HPC, it would lead to a contradiction, as this would be a winning play for Adam. Therefore, if Adam plays this way, i becomes eventually an HPC. Once this is the case, Adam can choose not to request i and play in the subgame $G_k^{\neg i}$, obtained from G_k by removing the states $\{i, j\}$ and i. $G_k^{\neg i}$ is isomorphic to G_{k-1}, and the projection of σ to it is winning. Proposition 9-β states that i will be an HPC (w.r.t. G_k) as long as the token does not leave $G_k^{\neg i}$. By induction hypothesis, Adam can force Eve to visit at least $(k-1)!$ different memory states in this subgame, all of them with i as an HPC. Finally, Proposition 9-γ implies that for different i, j, the two associated sets of memory states (of cardinal $(k-1)!$) are disjoint. This yields $k!$ different memory states for any winning strategy for G_k. The overall "memory strategy" for Adam is described below:

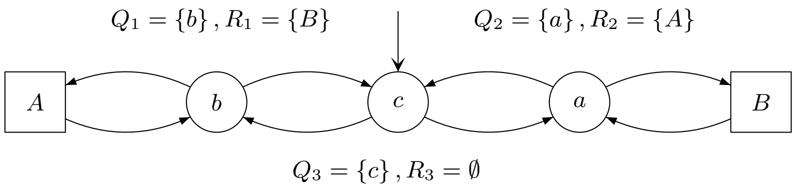
- Play according to the strategy τ_1 (requesting unanswered 1), until 1 is an HPC.
- Then play in the subgame without 1 the memory strategy for this game to force the memory to go through $(k-1)!$ different states. Condition 1 remains an HPC during this step.
- Repeat the same two steps for conditions $2, 3, \ldots, k$.

4 An algorithm to solve Streett games

The algorithm presented in this section is a specialized version of Zielonka's in [Zie98]. The point of this study is to get a better complexity than the IAR reduction, without relying on the practical behaviour of the Strategy Improvement in [VJ00]. Another interresting point is that this algorithm yields a positional strategy for the Rabin player. The existence of such a strategy is well known, yet the reduction through IAR gives memory-dependant strategies for Adam in simple cases.

4.1 Naïve Approach

A first idea is to consider a Streett game as a conjunction of $k!$ different parity games, each of them representing a hierarchy of importance between the basic Streett conditions. In this view, the parity condition for permutation π would be defined by: $\forall q, c_\pi(q) = \max\left\{2i \mid q \in R_{\pi_i}\right\} \cup \left\{2j - 1 \mid q \in Q_{\pi_j}\right\}$. A winning region for Adam in one of these games is winning for him in the Streett one.



$Q_1 = \{b\}, R_1 = \{B\}$
$Q_2 = \{a\}, R_2 = \{A\}$
A
b
c
a
B
$Q_3 = \{c\}, R_3 = \emptyset$

Fig. 3. A game where the naïve algorithm does not work

Unfortunately, the converse property for Eve does not hold: she may win each parity game and still lose the Streett game, as shown in figure 3.

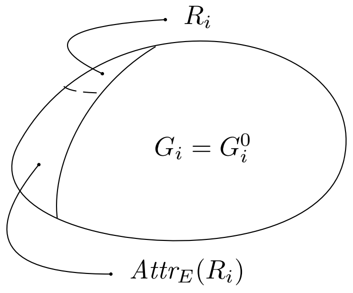
On this graph, Eve can win the games where (A, a) is more important than (B, b), by staying in the left part, and the one where (B, b) is more important than (A, a), by staying in the right part. The Streett game, however, is winning for Adam, as Eve must cross c to go from one part to the other. It proves that the choice of the priority between the colors cannot be made *a priori*: the algorithm in the next section reconsiders its choices for each subgame.

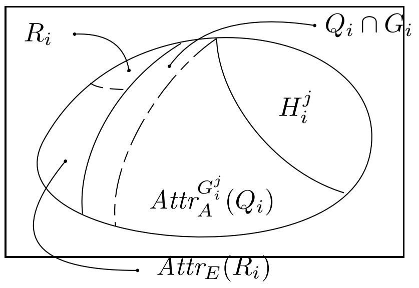
4.2 Algorithm

Idea: The main idea behind our algorithm is that Adam cannot win if Eve can visit the R-part of each of the basic Streett conditions. The first step is thus for Adam to choose a basic Streett condition, and avoid its R-part. In turn, Eve must now avoid the Q-part of the same condition in the remaining game. Once both of them did that, we have defined a new game, with one less basic condition. It is important to notice that *Adam* chooses the considered condition: if the computation ends favourably for Adam, then the region is really winning for him, while if Eve wins with this choice, Adam can change it. This algorithm has a time complexity of $\mathrm{O}(n^{2k}.k!)$: the n^{2k} factor of the complexity comes from two recursions on the number of states, while the $k!$ part comes from the ability of Adam to change its choice, which forces the algorithm to consider each of the basic conditions as the first one.

Description: Each main iteration of the algorithm determines a part of the winning region for Adam, and removes it from the game. This iterative process ends when no more region for Adam is found. We show then that Eve is the winner in the remaining region. We first give a description of the sub-algorithm used to compute winning regions for Adam, the proof of correctness will be made in the next section. This sub-algorithm will be run for each condition.

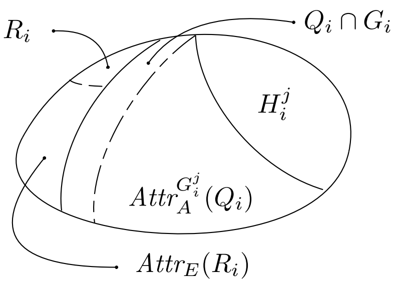
1. The set $Attr_E(R_i)$ is removed from the game until the end of the computation for this pair. This defines the subgame $G_i = G \setminus Attr_E(R_i)$, which is a trap for Eve. A decreasing sequence G_i^j of subgames of G_i will now be computed, with the property that all removed states are winning for Eve in G_i. The last G_i^j (when $G_i^j = G_i^{j+1}$) is the winning region for Adam in G.





Note that in G_i^j, Adam can win by visiting infinitely often Q_i. Hence, Eve's winning region cannot be included in $Attr_A^{G_i^j}(Q_i)$.

2. In the same way, a new subgame is defined: $H_i^j = G_i^j \setminus Attr_A^{G_i^j}(Q_i)$. H_i^j is a trap for Adam in G_i^j (but not in G)



3. The pair (Q_i, R_i) does not appear in H_i^j. The algorithm can thus be used recursively to compute the winning regions $W_E(H_i^j)$ and $W_A(H_i^j)$.
4. The winning region $W_E(H_i^j)$ for Eve and its G_i^j-attractor are removed from G_i^j. This defines G_i^{j+1}. By remark 5, G_i^{j+1} is still a trap for Eve in G. The algorithm restarts with $j = j+1$ at step 3, unless $G_i^{j+1} = G_i^j$.

Remark 10. All subgames defined by this algorithm are traps for either Eve or Adam. Remark 5 allow us to say that:

– The winning region for Adam in one of the G_i^j is still winning for him in G.
– The winning region for Eve in one of the H_i^j is still winning for her in G_i^j (and in G_i by construction).

When the loop ends either $G_i^j = G_i^{j+1} \neq \emptyset$ is winning for Adam (i.e., no states were removed), or $G_i^j = G_i^{j+1} = \emptyset$ (i.e., Eve wins everywhere in H_i^j). We give now an informal sketch of proof:

In the first case, the winning region for Eve in H_i^j is empty, hence G_i^j and $Attr_A(G_i^j)$ are winning for Adam in G (see Remark 10). This means that we found a winning region for Adam and we can rerun the whole algorithm on a smaller game (see figure 4). Note that all the conditions will be considered again, even the ones that had already been discarded (ending in case 2).

In the second case, all states in G_i^0 belong to a $W_E(H_i^j)$ or its attractor, so Eve wins everywhere in G_i^0 (Remark 10 again). The algorithm proceeds with the next condition. If none of the remaining conditions yields a winning region for Adam, Eve wins in the whole game (see figure 5).

4.3 Proof and Complexity

Validity of the resulting strategies Each iteration of the algorithm removes states of the game until either no state can be removed, or all the states have been removed.

Case 1: There exist i and j such that G_i^j is not empty and H_i^j is winning for Adam. Thus the game is in a form similar to the one in figure 4.

- $G_i^j \cap R_i = \emptyset$
- G_i^j is a trap for Eve
- $G_i^j = Attr_A^{G_i^j}(Q_i) \cup H_i^j$ with $W_A(H_i^j) = H_i^j$

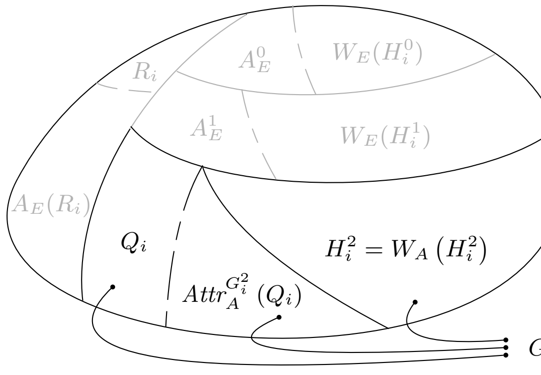


Fig. 4. Case 1: G_i^j is a winning region for Adam.
Here the sub-algorithm stopped at the third iteration, so $j = 2$

Adam wins everywhere in G_i^j with the following strategy[3]:

- In $Attr_A(Q_i)$, the attractor strategy.
- In H_i^j, the corresponding winning strategy for Adam.

Eve may then choose either to stay forever in H_i^j after a finite number of moves or to cross $Attr_A(Q_i)$ (and Q_i) an infinite number of times. Whatever her choice, Adam wins. By remark 10, G_i^j (and its attractor *in* G) is a winning region for Adam in G.

Case 2 : If no subgame H_i^j is fully winning for Adam, then Eve wins in the whole game: in this case, the game is partitioned as in figure 5 for *each* i.

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

[3] There are many winning strategies, but this one is the natural one corresponding to the construction of the set

- A_E^j stands for $Attr_E^{G_i^j}(W_E(H_i^j))$
- Eve wins if the token stays in one of the $W_E(H_i^j)$
- Adam can escape the A_E^j only by going up (or to $A_E(R_i)$)

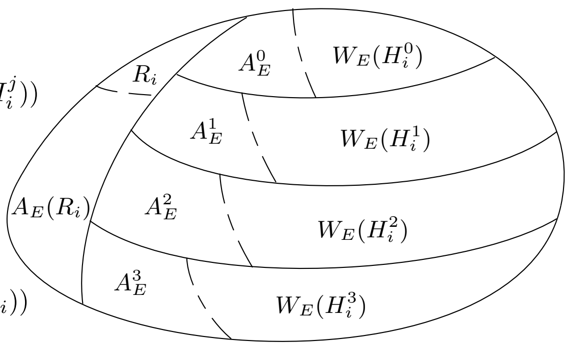


Fig. 5. Case 2: Eve wins everywhere in G_i
Here the subalgorithm stopped at the fourth iteration, so j ranges over $\{0 \dots 3\}$)

- In the subgames H_i^j: the subgame strategy (with $(k-1)!$ memory).
- In the attractors of the subgames: the attractor strategy.
- In $Attr_E(R_i)$: the attractor strategy.
- In R_i: Eve updates her top-level memory to $(i+1) \mod k$ and applies $\sigma_{i+1[k]}$.

The subgames' strategies and their attractors are not global strategies: Adam can escape a subgame or its attractor by jumping to the attractor of a subgame with a lower exposant, or to $Attr_E(R_i)$. But he has only two choices: either to eventually stay in one of the subgames, or to cross each of the R_i infinitely often. Whatever his choice, Eve wins everywhere in G.

Time complexity The time complexity for this algorithm is $O(k!n^{2k})$: the algorithm removes at least one state in each iteration, and calls n copies of itself with one less condition. The internal operations (*Attr* computation, mainly) are all polynomial.

4.4 Conclusion

The algorithm in section 4.2 gives the winning regions in better theoretical time: $O(n^k)$ instead of $O(n^{k^2})$. Adam's strategy is indeed positional, as his strategy is a compilation (regionwise) of attractor strategies. Finally, the new algorithm is much more natural, dealing directly with attractors and memory states instead of using a reduction to parity games. The drawback is in the comparison with the Strategy Improvment algorithm: there are families where our algorithm is exponential in n, while there are none yet for an algorithm using the IAR reduction and the Strategy Improvment algorithm.

5 Conclusion

We have shown that the need for factorial memory in Streett games does not depend on the exponential size of the graph. Our algorithm for Streett games presented a way to solve them without refering to algorithms for parity games.

Another interresting point is that it gives a way to get a positional strategy for the Rabin player that is more efficient than simply guessing it - though guessing the right permutation still highly speed up the process We intend to implement this algorithm in the Symprog program presented in [Wal03], to get comparison of time complexity with the IAR algorithm over real examples.

Acknowledgments. I wish to thank Wolfgang Thomas, for many fruitful discussions that led to these results, and Anca Muscholl, for her constant help in the presentation of this manuscript.

References

[BLV96] N. Buhrke, H. Lescow and J. Vöge. Strategy Construction in Infinite Games with Streett and Rabin Chain Winning Conditions. In *TACAS*, pages 207–224, 1996.

[DJW97] S. Dziembowski, M. Jurdziński and I. Walukiewicz. How Much Memory Is Needed to Win Infinite Games? In *LICS*, pages 99–110, 1997.

[EJ88] E. A. Emerson and C. S. Jutla. The Complexity of Tree Automata and Logics of Programs. In *FOCS*, pages 328–337, 1988

[EL85] E. A. Emerson and C.-L. Lei. Modalities for Model Checking: Branching Time Strikes Back. In *POPL*, pages 84–96, 1985

[Hun04] P. Hunter. co-NP-Completeness of Streett Games. Private communication, 2004

[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. *TCS*, volume 141(1-2), pages 69–107, 1995

[Saf92] S. Safra. Exponential Determinization for ω-Automata with Strong-Fairness Acceptance Condition. In *STOC*, pages 275–282, 1992

[Sch02] S. Schwoon. Determinization and Complementation of Streett Automata. In *ALIG*, volume 2500 of *Lecture Notes in Computer Science*, pages 79–91, 2002.

[Tho95] W. Thomas. On the Synthesis of Strategies in Infinite Games. In *STACS*, volume 900 of *LNCS*, pages 1–13, 1995.

[VJ00] J. Vöge and M. Jurdziński. A Discrete Strategy Improvement Algorithm for Solving Parity Games. In *CAV*, volume 1855 of *LNCS*, pages 202–215, 2000.

[Wal03] N. Wallmeier. *Symbolische Synthese zustandsbasierter reaktiver Programme*. Diploma thesis, RWTH Aachen, 2003.

[WHT03] N. Wallmeier, P. Hutten and W. Thomas. Symbolic Synthesis of Finite-State Controllers for Request-Response Specifications. In *CIAA*, volume 2759 of *LNCS*, pages 11–22, 2003.

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