On polynomial-size programs winning finite-state games *

Helmut Lescow

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
D-24098 Kiel
email: hel@informatik.uni-kiel.d400.de

Abstract. Finite-state reactive programs are identified with finite automata which realize winning strategies in Büchi-Landweber games. The games are specified by finite "game graphs" equipped with different winning conditions ("Rabin condition", "Streett condition" and "Muller condition", defined in analogy to the theory of ω-automata). We show that for two classes of games with Muller winning condition polynomials are both an upper and a lower bound for the size of winning reactive programs. Also we give a new proof for the existence of no-memory strategies in games with Rabin winning condition, as well as an exponential lower bound for games with Streett winning condition.

1 Introduction

In the construction of correct programs there are two complementary approaches: the verification of an existing program relative to a given specification (for example, by a model-checking algorithm), or the algorithmic synthesis of a correct program from the specification. The present paper deals with the second approach in the context of nonterminating finite-state reactive programs.

We study the problem within the terminology of infinite games. Here a specification describes a set of infinite computations (represented by a set L of ω-words), which is defined by a finite automaton on infinite words. A computation is an ω-word built up interactively by two parties, called player 0 (also called 'Control') and player 1 (also called 'Disturbance'). Thus a computation can be viewed as an infinite play in a game associated with L. The two players perform their actions in turn; and if a play built up in this way belongs to L then player 0 is said to win the play. A strategy for player 0 fixes the actions of this player based on the information which actions have been taken so far; and it is called a winning strategy if it ensures that such a play will be won by player 0 (whatever actions are taken by player 1). A reactive program is correct with respect to the specified ω-language L if it realizes a winning strategy for player 0. The strategy (or reactive program) is said to be finite-state if this realization is given by a

* This work was supported by Deutsche Forschungsgemeinschaft (Project Th 352/3-1).

finite automaton with output. The size of such a strategy is identified with the number of states (or memory) of the corresponding automaton.

The theoretical basis of this approach is the theorem of Büchi and Landweber [BL69] which says that from a Muller automaton (defining a set L of ω-words and thus an "infinite finite-state game") one can decide effectively whether player 0 has a winning strategy in the game associated with L and in this case construct even a finite-state winning strategy for player 0. The applicability of the Büchi-Landweber Theorem is questionable due to the high complexity of the strategy construction and by the large size of the automata which realize the winning strategies. As is well-known in the literature on infinite games ([GH82] [Bü83] [EJ91] [McN93]) the size (number of states) of a strategy automaton is bounded by the factorial of the number of states of the Muller automaton defining the game. In fact, there have been relatively few papers on applications of the game theoretical approach to the construction of correct reactive programs; among the sparse examples are the works of Abadi, Lamport and Wolper [ALW89] and Pnueli and Rosner [PR89].

The purpose of the present paper is to prove the first lower bounds on the size of strategy automata and to exhibit some special cases of game specifications (by ω-automata) where polynomials are both an upper and a lower bound for the size of strategy automata. This may be useful in finding out cases where an automatic construction and the implementation of finite-state strategies is efficient.

We refer to three kinds of acceptance condition in the ω-automata that specify infinite games (which are called "winning condition" in the context of games): the "Rabin condition", the "Streett condition", and the "Muller condition". After introducing these conditions we collect some technical prerequisites in Section 2. In Section 3 we show how to compose strategies. The main results on polynomial size strategies are given in Section 4 and Section 5. We show that polynomial size strategies can be guaranteed for games with Muller condition where the collection of accepting (or winning) loops in the game graph satisfies additional restrictions. We consider two such restrictions: that these loops form a chain (say of length k), and that the system of accepting loops is closed upwards w.r.t. to set inclusion, containing say k minimal loops. The main result states that (for game graphs with n nodes) the polynomial n^k provides both an upper and a lower bound for the size of winning strategies in these two cases. In Section 5 we also show that in general games with Muller winning condition may require an exponential memory. In Section 6 we consider Rabin and Streett conditions. We develop a proof technique to show that games with Rabin condition can be won by player 0 with a memoryless strategy (earlier proved by e.g. [Em85], [EJ91], [Kla94]). Subsequently we show that games with Streett condition may require a memory of exponential size. We conclude with some open questions.

I thank Wolfgang Thomas for many helpful discussions during the preparation of this paper.

2 Definitions and Preliminaries

For the treatment of infinite games, it is convenient to consider a type of "ω-automaton" introduced by McNaughton [McN93]. Since acceptance (and winning condition) depends only on visits of states, we abstract from the labels of transitions; moreover, each state is associated with just one of the two players (whose turn it is to make the next move). We use the following set-up:

A *finite-state game* is given by a bipartite finite directed graph G and a "winning condition". The idea is that two players, called player 0 and player 1, are moving a token alternatively from vertex to vertex along edges in the game graph. A game graph is presented in the form $G = (Q, Q_0, Q_1, E)$ with $Q = Q_0 \dot\cup Q_1$ and $E \subseteq (Q_0 \times Q_1) \cup (Q_1 \times Q_0)$. Q_i is the set of vertices where it is the turn of player i to move the token. Each vertex of the game graph has at least one outgoing edge. In graphical representations of game graphs we use circles for nodes in Q_0 and squares for those in Q_1.

A *play* π is an infinite sequence of nodes from Q visited by the token in successive moves, i.e., a sequence $\pi \in Q^\omega$ with $(\pi[i], \pi[i+1]) \in E$ for all i. A finite sequence over Q with this property is called *partial play*. For each play we declare a winner by a *winning (or accepting) condition* $Acc : Q^\omega \mapsto \{0, 1\}$ that maps a play π to i iff the play π is won by player i. So a game Γ is given by a pair $\Gamma = (G, Acc)$.

The games we want to discuss here are Muller games, Rabin games, and Streett games (referring to the analogous acceptance conditions for ω-automata, cf. [Tho90]). These games are characterized by special winning conditions. A *Muller condition* is induced by a system $\mathcal{F} \subseteq 2^Q$ of sets of nodes in the corresponding game graph, and the condition Acc defined by

$$Acc(\pi) = 0 \iff Inf(\pi) \in \mathcal{F}$$

where $Inf(\pi)$ is the set of nodes visited infinitely often during the play π. We call the sets in $\mathcal{F}$ positive loops (assuming without loss of generality these sets really form loops in the game graph). A *Rabin condition* is given by a system $\Omega \subseteq 2^Q \times 2^Q$, and the function Acc defined by $Acc(\pi) = 0$ iff

$$\exists (L, U) \in \Omega \text{ s.t. } Inf(\pi) \cap L = \emptyset \ \wedge \ Inf(\pi) \cap U \neq \emptyset$$

The *Streett condition* is dual to the Rabin condition in the sense that a play π is won by player 0 in a Rabin game iff π is accepted for player 1 in the corresponding Streett game. So the Streett acceptance condition is given by a system $\Omega \subseteq 2^Q \times 2^Q$ where the function Acc is defined by $Acc(\pi) = 0$ iff

$$\forall (L, U) \in \Omega : \ Inf(\pi) \cap U \neq \emptyset \ \rightarrow \ Inf(\pi) \cap L \neq \emptyset$$

A *strategy for player i* in the game Γ is a function which associates with a partial play ending in Q_i a node in Q_{1-i}.W.l.o.g. a strategy may be defined as a partial function $\sigma : Q^* Q_i \mapsto Q_{1-i}$ with $\sigma(p_1, \ldots, p_k) = p_{k+1}$ such that $(p_k, p_{k+1}) \in E$. The strategy σ is a *winning strategy* if, for any choice of moves of player $1 - i$,

it induces a play won by player i. If a winning strategy for player i in the game Γ exists we say *player i wins Γ*.

A strategy uses a partial play for deciding what will be the next move. Büchi and Landweber showed that finite automata (with fixed finite memory of partial plays) are sufficient to realize winning strategies for games on finite graphs (cf. [BL69]), i.e. the strategies can be built up by the game graph and this additional memory M providing each node of the graph with the information necessary for choosing the next move. To realize the strategy in terms of graphs we use a notion of strategy graph that is equivalent to using finite automata with output, but has the advantage that the underlying structure of the game graph is preserved. So a strategy σ for player i with memory M in $\Gamma = (G, Acc)$ can be realized as a *strategy graph* $\mathfrak{A}_\sigma = (Q_\sigma, E_\sigma)$ with

(a) $Q_\sigma = Q \times M$

(b) $\forall (q,m)\ \big(q \in Q_i \rightarrow \exists^1\ (q',m')\big[(q,q') \in E \wedge \big((q,m),(q',m')\big) \in E_\sigma\big]\big)$
(For all pairs of a vertex q in Q_i and a memory content m the new vertex q' and updated memory content m' are unique.)

(c) $\forall (q,m)\ \forall\ (q,q') \in E\ \big[q \in Q_{1-i} \rightarrow \exists^1 m'\big((q,m),(q',m')\big) \in E_\sigma\big]$
(For all pairs of a vertex q in Q_{1-i} and a memory content m, and for all choices of q' as successor of q in G the updated memory content m' is uniquely given.)

We call a strategy for player i *no-memory* if no additional memory is required, i.e. the strategy graph can be achieved by deleting all but one outgoing edges of the nodes in Q_i of the game graph.

If we are interested only in the first components of states in the strategy graph we will use the projection $Pr_1((q,m))$. We use the same notation for the projection on plays and sets. For a play $\pi = (p_1,m_1)\ (p_2,m_2),\ldots$ in the strategy graph we write $Pr_1(\pi)$ instead of $Pr_1((p_1,m_1))\ Pr_1((p_2,m_2)),\ldots$ and analogously, for sets T of nodes, $Pr_1(T)$ instead of $\{Pr_1(q) |\ q \in T\}$.

3 Basic strategies and their composition

Many strategies are obtained in a simple way by composing simpler strategies. In this section we explain the composition of strategy graphs and define some basic strategies to be used later.

Assume we have two strategy graphs $\mathfrak{A}_{\sigma_1} = (Q_{\sigma_1}, E_{\sigma_1})$ and $\mathfrak{A}_{\sigma_2} = (Q_{\sigma_2}, E_{\sigma_2})$ (realizing strategies for player i) in a game on the graph $G = (Q, Q_0, Q_1, E)$. We get the strategy graph for the strategy σ "Play first strategy $\mathfrak{A}_{\sigma_1}$ and then $\mathfrak{A}_{\sigma_2}$" by the following modifications in the strategy graphs $\mathfrak{A}_{\sigma_1}$ and $\mathfrak{A}_{\sigma_2}$: for all nodes $q \in Q_{\sigma_1}$ with $Pr_1(q) \in Pr_1(Q_{\sigma_2})$ remove q and all its outgoing edges, and replace any edge $(p,q) \in E_1$ by an edge (p,q') for some $q' \in Q_{\sigma_2}$ with $Pr_1(q) = Pr_1(q')$. To meet condition c) for strategy graphs we may have to add some edges: If there are nodes $p,q \in Pr_1(Q_{\sigma_1} \cup Q_{\sigma_2})$, $p \in Q_{1-i}$, an edge $(p,q) \in E$, and a node $p' \in Q_{\sigma_1} \cup Q_{\sigma_2}$ such that for all edges $(p',q') \in E_\sigma$ we

have $Pr_1(p') = p \Rightarrow Pr_1(q') \neq q$ then we add an edge (p', q') for an arbitrary node $q' \in Q_{\sigma_1} \cup Q_{\sigma_2}$ with $Pr_1(q') \neq q$.

Let us give some basic strategies that we will use later for composing new ones. Here we refer to a give set $T \subseteq Q$ equipped with a "rank function" (in our case a map into $\mathbb{N}$). The strategies are "Avoid T" (or "Keep out of T") and "Decrease the rank" (as introduced by Gurevich and Harrington in [GH82]).

To explain this in more detail we follow Zielonka [Zie94] and define an i-trap as a set $M \subseteq Q$ with the properties $\forall p \in Q_i \cap M \;\; \forall (p, p') \in E \;\; p' \in M$ and $\forall p \in Q_{1-i} \cap M \;\; \exists (p, p') \in E \;\; p' \in M$. Player i cannot leave M and player $1 - i$ cannot be forced to do so. If M is a $1 - i$-trap we can construct a strategy graph for player i to avoid nodes in $Q \setminus M$. For every node p in $M \cap Q_i$ there is an edge $e_p \in E$ that does not leave M. So we get the strategy graph for "Avoid $Q \setminus M$" by deleting for all $p \in Q_i$ the outgoing edges except e_p.

The standard way to fix the rank in the definition of "Decrease" the following sets U_i, which collect the states of Q from which player 0 can force a visit within i steps:

$$U_1 = \{p \in Q_0 \mid \exists (p, q) \in E \; q \in T\} \cup \{p \in Q_1 \mid \forall (p, q) \in E \; q \in T\}$$
$$U_{i+1} = U_i \cup \{p \in Q_0 \mid \exists (p, q) \in E \; q \in U_i\} \cup \{p \in Q_1 \mid \forall (p, q) \in E \; q \in U_i(T)\}.$$

Let $\mathrm{rank}(p) = r$ if $p \in U_r \setminus U_{r-1}$. For $p \in Q_i$ let $e_p = (p, q)$ be an edge with $\mathrm{rank}(p) > \mathrm{rank}(q)$. We get the strategy graph for "Decrease the rank" for player i by deleting all edges but $e_p \in E$ for $p \in Q_i$.

We shall apply the above construction only in cases where the result is indeed a strategy graph (i.e. that for any node an outgoing edge exists).

In the following lemmas we describe a special usage of the strategies "Avoid" and "Decrease".

Obviously the increasing sequence (U_i) becomes constant at some k since the game graph is finite. We will denote this U_k by *Force*(T).

Lemma 1. *(a) For every node in Force(T) player 0 has a no-memory strategy ("Decrease the rank") to force the token into T.*

(b) For every node in $Q \setminus$ Force(T) player 1 has a no-memory strategy to avoid T ("Avoid Force(T)").

The proof is easy and rests on the fact that for each node in *Force*(T) the rank will decrease until T is reached and that $Q \setminus$ *Force*(T) is a 0-trap.

To compose other strategies we introduce sub-graphs and sub-games. For a set $M \subseteq Q$ let $G_M = (M, E \cap (M \times M))$ be the sub-graph induced by M. For $M \subseteq Q$ we define $Leave_G(M)$ to be the set of targets of transitions in G which player 1 can choose for leaving M, i.e. $Leave_G(M) = \{q \in Q \setminus M \mid \exists p \in M \cap Q_1 \; (p, q) \in E\}$. If the game graph is clear we omit the index G.

Assuming player 0 has a winning strategy for the game (G_M, Acc), player 1 can avoid loosing only by moving the token to a node in $Leave(M)$. We indicate this situation by saying that $Leave(M)$ is "weakly forced" by player 0 (just disregarding the trivial case that the play remains in M).

Formally, for $Leave(M) \neq \emptyset$ we will define when T can be "weakly forced" by player 0 from nodes in M. Let

$$V_0 = T,$$
$$W_{i+1} = Force(V_i),$$
$$(*)\quad V_{i+1} = W_{i+1} \cup \bigcup \{M \subseteq Q \mid Leave(M) \subseteq W_{i+1} \wedge \text{ player 0 wins } (G_M, Acc)\}.$$

Again the sequence (W_i) is increasing and becomes constant at some k, and we write $WForce(T)$ for such W_k.

Lemma 2. *(a) For every node in $WForce(T)$ player 0 has a strategy to weakly force the token into T.*
(b) For every node in $Q \setminus WForce(T)$ player 1 has a no-memory strategy to avoid T.

Proof. For (a) let $p \in WForce(T)$. For a node in $W_i \setminus V_{i-1}$ player 0 has a no-memory strategy σ_i to force the token into V_{i-1} as stated in Lemma 1. And for a node in $V_i \setminus W_i$ we have a set M and a winning strategy σ_i' in (G_M, Acc) such that finally W_i is reached or player 0 will win the play by staying in the subset M. We achieve a strategy for player 0 to force the token weakly into T by the strategy "Use first σ_{i+1}, then σ_i', and then σ_i" which is composed as explained above, and which uses the the union of memories used by the strategies for the corresponding sub-games (G_M, Acc) in $(*)$.

For (b), $Q \setminus WForce(T)$ is a 0-trap, so the no-memory strategy for player 1 to avoid T can be introduced as in Lemma 1: "Avoid nodes in $WForce(T)$". □

For $M \subseteq Q$ we will write $Force_M(T)$, resp. $WForce_M(T)$, to indicate that the token can be forced (weakly forced, resp.) into T in the sub-game (G_M, Acc) induced by M.

4 Upper bounds for strategies in Muller games

Theorem 3. *(a) In a finite-state game (over a set of n states) with Muller winning condition where the positive loops form a chain of length k (by set inclusion), player 0 has a winning strategy of size n^k if he wins.*
(b) In the games of (a), player 1 has a no-memory strategy if he wins.

Proof. The idea for part (a) is to use a memory $M = F_1 \cup \times \ldots \times F_k$ where the F_i are the " positive loops" (in the specified set $\mathcal{F}$ of subsets of Q). A memory-entry $(p_1, \ldots, p_k)$ indicates that p_i is the momentary goal to be reached in F_i. We use here a hierarchy of goals: The goals $p_i \in F_i$ are used as a means to reach a goal $p_{i+1} \in F_{i+1}$ by weakly forcing it.

Formally, we build the desired polynomial size strategy inductively starting with the minimal positive loop in the given chain of positive loops.

Case $k = 1$: Let $P = (p_1, \ldots, p_m)$ be a permutation of the nodes in F_1. In this permutation we call p_{i+1} the P-successor of p_i and p_1 the P-successor of

p_m. We determine for each p_i the set $Force_{F_1}(p_i)$ which gives us a no-memory strategy for nodes in the set to force the token into p_i. If q is the P-successor of p in the permutation, then $p \in Force_{F_1}(q)$ must hold. Otherwise player 1 has a no-memory strategy to win in G_{F_1} by avoiding q (see part (b)). So we have for a node p a no-memory strategy σ_q to force the token to its P-successor q. Hence for the winning strategy composed by executing these strategies one after the other (execute σ_q after σ_p if q is the P-successor of p) a memory of size $|F_1|$ is sufficient.

Case $k > 1$: Again we use a permutation $P = (p_1, \ldots, p_m)$ of the nodes in F_k. For each node $q \in F_k$ we determine the set $WForce_{F_k}(q)$. For the P-successor q of p, $p \in WForce_{F_k}(q)$ must hold. Let σ_q be the strategy for weakly forcing the token from p to q using a memory M_q By the induction hypothesis we have that for all σ_q the memory size $|M_q| \leq n^{k-1}$. The desired strategy σ for player 0 uses σ_q after σ_p if q is the P-successor of p. The memory M used in σ is the union of memories used in σ_q for $q \in F_k$. Since $|F_k| \leq n$ holds composing the strategies yields a strategy with a memory of size $\leq n^k$.

Remark. If $Leave(F_i) = \emptyset$ and σ is a winning strategy for G_{F_i} then σ is also a winning strategy in G. If $Leave(F_i) \neq \emptyset$ this is only a "local" strategy and player 1 has a no-memory strategy to avoid loosing in F_i by leaving F_i.

For part (b) we show that if the above construction does not yield a strategy for player 0 the opponent has a no-memory winning strategy. Assuming that our construction did not give a winning strategy for player 0 there must exist $p, q \in F_k$ with $p \notin WForce_{F_k}(q)$. For all $j < k$ $Leave(F_j) \neq \emptyset$ or player 1 has a no-memory winning strategy in G_{F_j}. We distinguish two cases:
i) There is no $F_r \subseteq F_k \setminus WForce(q)$: Then the winning strategy for player 1 is:"Avoid $WForce_{F_k}(q)$". Using this no-memory strategy player 1 wins since no set $F_i \in \mathcal{F}$ is a subset of $F_k \setminus WForce_{F_k}(q)$.
ii) F_r is maximal positive loop in $F_k \setminus WForce_{F_k}(q)$: Here an arbitrary strategy "Avoid $WForce_{F_k}(q)$" is not sufficient since player 1 could not ensure that $F_i \not\subseteq Inf(\pi)$ for $i \leq r$. So we construct a special Avoid strategy:

This Avoid strategy is composed from three components, an arbitrary Avoid strategy to avoid nodes in $F_k \setminus F_r$, a strategy to leave loops F_j with $s < j \leq r$, and a no-memory winning strategy for player 1 in the subgame induced by F_s where F_s is the maximal positive loop $F_s \subseteq F_r$ such that player 1 has a no-memory strategy in (G_{F_s}, Acc) (if such an F_s exists).

The last strategy guarantees that player 0 cannot win in loops F_j with $j \leq s$, the "Leave-strategy" guarantees the same for loops F_j with $s < j \leq r$, and the first mentioned Avoid strategy does it for loops F_j with $j > r$. □

In the second part of this section we consider a modified situation, where the positive loops do not necessarily form a chain. A certain fixed number k of minimal positive loops is allowed, and any loop extending (by set inclusion) a positive loop is again positive. On the other hand, above any positive loop there are only loops which are again positive. We call this subclass of Muller games

M_δ. This notation is motivated by Landweber's Theorem: A regular ω-language L is in the Borel class G_δ iff it is recognizable by a Muller automaton with a system of accepting sets F_i which is superset closed (see [Lan69] or [Tho90]).

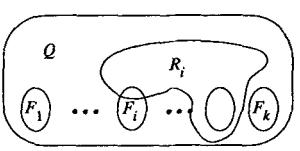
Theorem 4. *(a) In a finite state game $\Gamma \in M_\delta$ given with a Muller condition having precisely k minimal positive loops and a game graph G with n nodes, player 0 wins by a strategy of size n^k if he wins at all.*
(b) In a finite state game of (a) player 1 has a no-memory winning strategy if he wins.

Corollary 5. *Player 1 has a no-memory strategy in Muller games with a system of superset closed accepting sets (since the number of minimal positive loops is always finite).*

Proof of Theorem. For the proof of part (a) we give an algorithm for the construction of a winning strategy for player 0. If the algorithm produces a winning strategy for player 0 then it is one of size $\prod_{i=1}^{k} |F_i|$ where $F_1, \ldots, F_k$ are the minimal positive loops. Otherwise we can construct a winning strategy for player 1. Again the idea is to use a memory $M = F_1 \times \cdots \times F_k$ for minimal positive loops F_i with $(p_1, \ldots, p_k)$ indicating that p_i is the momentary goal in F_i. Note that player 0 has to ensure only that one of these F_i sets is included in $Inf(\pi)$. It is not important which other nodes are visited infinitely often since the system $\mathcal{F}$ is superset closed.

Construction of a polynomial size strategy
Let $I = \{1, \ldots, k\}$. For all $m \in \mathsf{X}_{i \in I} F_i$ determine $Force(T_m)$ where T_m is the set of nodes in the vector m. If $\bigcup_{i \in I} F_i \not\subseteq Force(T_m)$ then there is a node $p \in F_j$ from which player 1 has a no-memory strategy to avoid T_m. Delete j from I and repeat this until $\bigcup_{i \in I} F_i \subseteq Force(T_m)$. W.l.o.g. we assume that we were successful for $I = \{1, \ldots, k\}$ (otherwise, as we shall explain later, we get an even simpler strategy). Then for every m we have a no-memory strategy σ_m to force the token into T_m from nodes in $Force(T_m)$. Let $P_i = (p_{i,1}, \ldots, p_{i,k_i})$ be a permutation of the nodes in F_i and $m = (q_1, \ldots, q_k)$ be a memory content. Then player 0 uses the no-memory strategy σ_m until a node q_i in T_m is reached. The new memory content is $m' = (q_1, \ldots, q_{i-1}, q_i', q_{i+1}, \ldots, q_k)$ if q_i' is the P_i-successor of q_i in the permutation of F_i and player 0 uses the strategy $\sigma_{m'}$ until another momentary goal is reached.

This yields obviously a winning strategy for player 0. The size of the memory used is $\prod_{i=1}^{k} |F_i|$. (Therefore the strategy would be simpler for a smaller set I.) For part (b) our aim is to construct a sequence of sets R_i that are 0-traps and do not include a complete set F_j for $j \leq i$ but include at least one node from F_i. (R_i may include a set F_j with $j > i$.)



The sequence of 0-traps will provide player 1 with a sequence of no-memory strategies with which (executing one after the other) he wins the game.

In more detail, assume that by (a) we did not find a winning strategy for player 0. W.l.o.g. we assume the indexes are deleted from I in the order

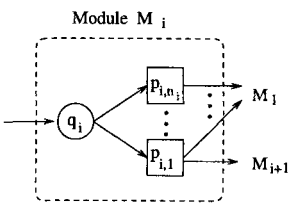
$k, k-1, \ldots, 1$. In every step there is a $m_j \in \mathsf{X}_{i \in I_j} F_i$ and a $q \in \bigcup_{i \in I_j} F_i$ with $q \notin Force(T_{m_j})$, $I_j = \{1, \ldots, j\}$. Let $R_j = Q \setminus Force(T_m)$. Let σ_{R_j} be "Avoid $Force(T_{m_j})$". So the composed strategy for player 1 is: "execute $\sigma_{R_1}, \ldots, \sigma_{R_k}$ in this order".

This is a winning strategy since staying in R_j does not allow player 0 to win in sets F_i with $i \leq j$. At least one node in F_i is in R_{i+1}. So player 0 has to enter R_{i+1} if he tries to win by exhausting a positive loop F_i. But finally having arrived in R_k, player 1 can ensure that no positive loop will be exhausted in $Inf(\pi)$. □

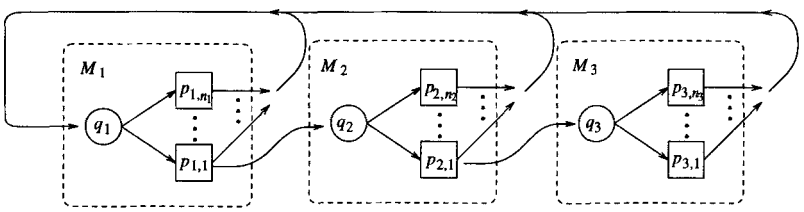
5 Lower bounds for strategies in Muller games

Theorem 6. *There is a family of finite-state games Γ_k, where Γ_k is given with a Muller winning condition consisting of a chain of positive loops of length k, such that player 0 wins Γ_k, however only by winning strategies of size $n^{\Omega(k)}$ where n is the number of states in Γ_k.*

Proof. Consider a game graph G_k, built up from k different subgraphs, here called modules, of the following kind:
Each module M_i has one node q_i in Q_0 and nodes $p_{i,1}, \ldots, p_{i,n_i}$ in Q_1. For all $p_{i,j}$ there exists an edge $(q_i, p_{i,j})$ and an edge $(p_{i,j}, q_1)$ to the first module M_1. For the node $p_{i,1}$ there is an edge $(p_{i,1}, q_{i+1})$ to the next module M_{i+1} (if there is one in G_n). We denote by P_j the nodes in the module M_j (including q_j).



For $k = 3$ we get the following graph:



The Muller condition is given by k positive loops, each of them defined by $F_i = \bigcup_{j=1}^{i} P_j$.

The core of the proof is by the following lemma:

Lemma 7. *Every winning strategy for player 0 in the game (G_k, Acc) (composed from modules M_i as given above with n_i nodes in Q_1) has at least a memory of size $\prod_{i=1}^{k} n_i$.*

The proof of Lemma 7 is done by induction on the memory used in the subgames (G_i, Acc), and rests on the fact that the strategy graph for (G_{i+1}, Acc) has to contain n_{i+1} copies of the strategy graph for (G_i, Acc).

To finish the proof of Theorem 6, assume that the game graph G_k is constructed as above. We take a module M_1 with $n_1 = 2^k$ nodes in Q_1, and in

general M_i with 2^{k+1-i} nodes in Q_1. So the graph G_k has $n < 2n_1 + k < 3n_1$ nodes since $k \leq log(n_1)$. Hence using the lemma we have

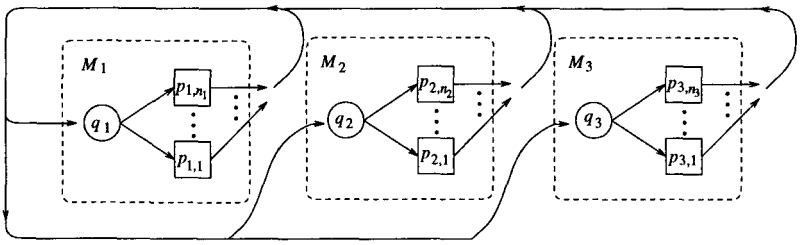
$$|M| = \prod_{i=1}^{k} n_i = \prod_{i=1}^{k} \frac{n_1}{2^i} = \frac{{n_1}^k}{2^{k+1}} = \frac{1}{2} \cdot {n_1}^{k-1} \in n^{\Omega(k)}$$

□

We use the same idea for Muller games with superset closed acceptance sets.

Theorem 8. *There is a family of games $\Gamma_k \in M_\delta$, where Γ_k is given with a Muller winning condition consisting of k minimal positive loops, such that player 0 wins Γ_k, however only by winning strategies of size $n^{\Omega(k)}$ where n is the number of nodes in Γ_k.*

Proof. We use modules as before which are connected in a modified way as indicated in the following figure (again for $k = 3$). There are additional edges in each module: for all $p_{i,j}$ and all q_l there is an edge $(p_{i,j}, q_l)$.



The Muller condition is given by k minimal positive loops each defined by $F_i = P_i$ and all possible unions of different F_i sets, i.e.

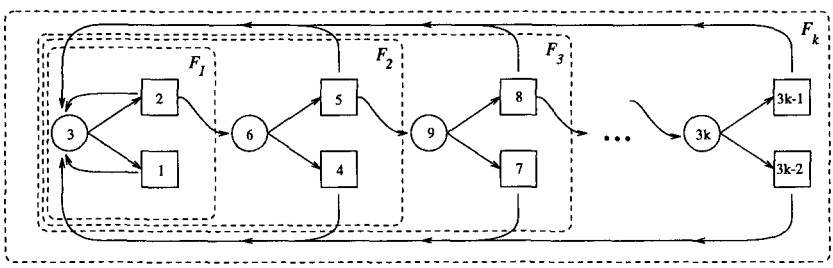
$$\mathcal{F} = \{F | F = \bigcup_{i \in I} P_i, \quad I \subseteq \{1, \ldots, k\}\}.$$

As before it is easy to show:

Lemma 9. *Every winning strategy for player 0 in the game G_k has at least a memory of size $\prod_{i=1}^{k} n_i$*

The proof of the theorem is now as in the preceeding case. □

Le us finally show an exponential lower bound for strategies in unrestricted Muller games. Define the games Γ_k with modules as in Theorem 6, where each module is of size 3:



Theorem 10. *Player 0 wins Γ_k, and any finite state winning strategy of player 0 has to use at least a memory of size $2^{\Omega(n)}$ if n is the number of vertices in G_k.*

Proof. An application of Lemma 7 shows that we need a memory of size 2^k, and since $n = 3k$, we have $|M| = 2^{n/3}$. □

Let us remark that this lower bound involves only games whose winning condition consists of a number of positive loops that increases linearly in the size of the game graph. This is an improvement on a result of S. Seibert [Sei95] which shows the above theorem using example games which involve Muller conditions of exponential size in n.

6 Rabin and Streett games

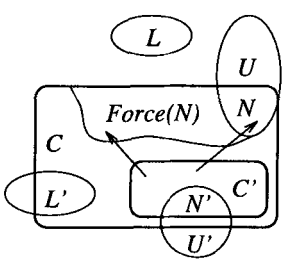
In the previous sections we examined Muller games. Now we want to show some upper and lower bounds on strategies for Rabin and Streett games. It is known that games with Rabin winning condition allow memoryless winning strategies for player 0 ([Em85], [Kla94], [MPS95]). We reprove the result here in a new form and show that the opponent may have to use memory of exponential size. (This is in contrast to the restricted "Rabin chain condition" or "parity condition", where the respective winner always has a no-memory winning strategy [EJ91], [Tho95].)

Theorem 11. *(a) In a finite-state game with Rabin winning condition, player 0 has a no-memory strategy if he wins.*
(b) The opponent may have to use a strategy with memory if he wins (in fact, a strategy with memory of exponential size in the size of the game graph).

Corollary 12. *In a finite-state game with Streett winning condition, player 1 has a no-memory strategy if he wins, whereas player 0 may have to use a strategy with memory of exponential size.*

For the proof of a) we assume that player 0 has a winning strategy with memory in the Rabin game and reduce it in several steps to a no-memory strategy. Each step uses Lemma 13 below that guarantees that at least one node does not need the additional memory.

The reduction of the memory is based on the following idea: We consider a strongly connected component C in the strategy graph for player 0 that does not have any outgoing edges. Since the strategy graph is assumed to be a winning strategy for player 0, the set $Pr_1(C)$ must be compatible with an accepting pair (L, U), i.e. we have $Pr_1(C) \cap L = \emptyset$ and $Pr_1(C) \cap U \neq \emptyset$.



Since it is sufficient to visit one of the nodes in $Pr_1(C) \cap U$ infinitely often and there is no escape from C only one node with this first component is necessary.

By disregarding these nodes in C we get other smaller components (in the figure: C') to which we apply the same elimination procedure if leaving the component is only possible by visiting a node in $Pr_1(C) \cap U$.

Let $N \subseteq Q$. A strongly connected component C is *maximal w.r.t.* N if there is no strongly connected component $C' \supset C$ such that $C' \cap N = \emptyset$. We call a maximal strongly connected component C a *latest component* if all paths from C to other maximal components include nodes from N.

Lemma 13 [Reduction Lemma]. *Let $\mathfrak{A}_\sigma$ be the graph of a winning strategy for player 0 in a Rabin game with the accepting pairs $(L_1, U_1), \ldots, (L_k, U_k)$. Let $N \subseteq Q_\sigma$. Furthermore let C be a latest component in $\mathfrak{A}_\sigma$ such that*

- *for every path S from C to another component C' there is a pair (L, U) with $Pr_1(S) \cap L = \emptyset \wedge Pr_1(S) \cap N \cap U \neq \emptyset$;*
- *if there is a path from C to another maximal component C' then there is a path from C' to C.*

Then there is a node $(q, m) \in C$ that can be replaced by the node q with some edges to and from nodes in $\mathfrak{A}_\sigma$, all nodes (q, m') can be removed from $\mathfrak{A}_\sigma$, and the resulting strategy graph determines again a winning strategy.

Proof of the Reduction Lemma. Assume the requirements are met. Since $\mathfrak{A}_\sigma$ is a winning strategy, $Pr_1(C)$ must be a set of vertices compatible with a pair (L_i, U_i) for some $i \in \{1, \ldots, k\}$. Let I_C be the set of indices of pairs compatible with C,

$$I_C = \{i \mid L_i \cap Pr_1(C) = \emptyset \wedge U_i \cap Pr_1(C) \neq \emptyset\}$$

Since there has to be at least one non-empty set U_i with $i \in I_C$ the set $N' = \bigcup_{i \in I_C} U_i \cap Pr_1(C)$ is not empty.

Now we get a new strategy graph $\mathfrak{A}'_\sigma$ by the following steps: For any $q \in N'$ add q to $\mathfrak{A}_\sigma$. Remove all nodes (q, m') from $\mathfrak{A}_\sigma$ and replace all edges to nodes (q, m') by edges to q.

It is easy to check that $\mathfrak{A}'_\sigma$ is still a winning strategy for player 0 in the game. □

Proof of Theorem 11. For a) assume we have a winning strategy graph $\mathfrak{A}_\sigma$ for the Rabin game. Let C be a latest component of $\mathfrak{A}_\sigma$. In the first step set $N = \emptyset$. Using the Reduction Lemma we find nodes in C that can be replaced by "no-memory nodes". We add the no-memory nodes to N. If C is not completely converted into no-memory nodes then C without the reduced nodes consists of several strongly connected components $C_1, \ldots C_k$. By construction of N every path from C_i to C_j through N is compatible with an accepting pair. Hence we can (recursively) apply the Reduction Lemma until no components of non-reduced nodes exist. So C (and finally $\mathfrak{A}_\sigma$) are completely reduced such that we get a no-memory strategy.

For b) it suffices to show that player 0 has to use a strategy of exponential size in a Streett game. The idea is to give a family of Streett games that is equivalent to the family of Muller games of Theorem 10 above, and then apply that theorem.

So we take the same game graph G_k as is Theorem 10 and the Streett winning condition given by

$$i, i+1 \in Q_1 \Rightarrow (\{i\}, \{i+1\}) \in \Omega \wedge (\{i+1\}, \{i\}) \in \Omega$$

using the numbering of states as in the figure of Theorem 10.

It is easy to show that player 0 wins a play π in the Muller game iff he wins π in the Streett game. Thus the games are equivalent and an application of Theorem 10 shows that a winning strategy uses at least a memory of size $2^k = 2^{|Q|/3}$. □

7 Conclusion

The games allowing polynomial-size winning strategies as considered above show a common feature, namely that only one or a fixed number of winning loops in a game graph are minimal w.r.t. set inclusion. Present work is directed to a general result which establishes polynomial-size strategies for any game with this property.

Concerning arbitrary "superset closed" winning conditions, it remains to be analysed whether polynomial-size strategies are possible without a bound k on the number of minimal positive loops.

In this paper we gave some algorithms for effective construction of strategies. But these constructions are by no means efficient. Indeed, the given constructions require all exponential time in the size of the game graph. So an interesting question is to find classes of games for which the *construction of strategies* is polynomial, not only the size of the resulting strategy.

References

[ALW89] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In G. Ausiello et al., editor, *Automata, Languages, and Programming*, volume 372 of *LNCS*, pages 1 – 17, Berlin, Heidelberg, New York, 1989. Springer-Verlag.

[BL69] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.*, 138:295 – 311, 1969.

[Bü83] J. R. Büchi. State strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$. *J. Symb. Logic*, 48:1171 – 1198, 1983.

[EJ91] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. 32nd IEEE Symp. on the Foundations of Computing*, pages 368 – 377, 1991.

[Em85] E. A. Emerson. Automat, tableaux, and temporal logics. In G. Goos and J. Hartmanis, editors, *Logics of Programs*, volume 803 of *LNCS*, pages 79 – 88, Berlin, Heidelberg, New York, 1985. Springer-Verlag.

[GH82] Y. Gurevich and L. Harrinton. Trees, automata, and games. In *Proc 14th ACM Symp. on the Theory of computing*, pages 60 –65, San Fancisco, 1982.

[Kla94] N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure and Appl. Math.*, 69:243 – 268, 1994.

[Lan69] L. H. Landweber. Decision problems for ω-automata. *Mathematical Systems Theory*, 3:376 – 384, 1969.

[McN93] R. McNaughton. Infinite games played on infinite graphs. *Ann. Pure Appl Logic*, 65:149 – 184, 1993.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed-systems. In Ernst W. Mayr and Claude Puech, editors, *STACS 95*, pages 229 – 242, Berlin, Heidelberg, New-York, 1995. Springer-Verlag.

[PR89] A. Pnueli and R. Rosner. On the systhesis of a reactive module. In *Proc. 16th ACM Sympos. on Principles of Prog. Lang.*, pages 179 – 190, Austin, 1989.

[Sei95] S. Seibert. PhD thesis, University of Kiel, 1995. (in preparation).

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 131–191. North-Holland, Amsterdam, 1990.

[Tho95] W. Thomas. On effective strategies in infinite games. In Ernst W. Mayr and Claude Puech, editors, *STACS 95*, pages 1 – 13, Berlin, Heidelberg, New-York, 1995. Springer-Verlag.

[Zie94] W. Zielonka. Infinite games on finitely coloured graphs with some applications. *Manuscript*, 1994.