On the synthesis of strategies in infinite games *

Wolfgang Thomas

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel, D-24098 Kiel
`wt@informatik.uni-kiel.d400.de`

Abstract. Infinite two-person games are a natural framework for the study of reactive nonterminating programs. The effective construction of winning strategies in such games is an approach to the synthesis of reactive programs. We describe the automata theoretic setting of infinite games (given by "game graphs"), outline a new construction of winning strategies in finite-state games, and formulate some questions which arise for games over effectively presented infinite graphs.

1 Introduction

One of the origins of automata theory over infinite strings was the interest in verifying and synthesizing switching circuits. These circuits were considered as transforming infinite input sequences into output sequences, and systems of restricted arithmetic served as specification formalisms ([Ch63]). With Büchi's decision procedure for the monadic second-order theory S1S of one successor ([Bü62]), it turned out that the "solution problem" (in more recent terminology: the verification problem or model checking problem) for circuits with respect to specifications in S1S was settled. Büchi's proof showed that S1S specifications can be turned into ω-automata, whence the verification problem amounts to an inclusion test for ω-languages defined by automata. In the context of nonterminating reactive (finite-state) programs, this result was refined and extended in many ways during the past decade, especially for several systems of temporal logic (in place of S1S), and with the aim of obtaining more efficient decision procedures for program verification. See [CGL94] for a survey of the state-of-the-art.

However, this approach does not fully exploit the available automata theoretic results: Büchi and Landweber presented in their fundamental work [BL69] an algorithm which decides the realizability of a given specification and in this case synthesizes a circuit (or finite-state reactive program) from the specification. This result is "better" than the decidability of S1S in the sense that it is better to automatically construct a correct reactive program than to verify an existing one.

---

* This work was supported by the ESPRIT Basic Research Action 6317 ASMICS ("Algebraic and Syntactic Methods in Computer Science")

One may consider a nonterminating reactive program as a player in a two-person game against the "environment" (the second player). A play of the game is an infinite sequence of actions performed in alternation by the two players. The decision who wins is provided by a set S of plays (given by the specification, and containing plays with certain desirable properties). If for any choice of actions by the environment the program builds up a play in S, it is "correct" with respect to S. This approach was pursued in [ALW89], [PR89], among others (for more background and references we recommend [NYY92a]). However, the theory of infinite games still lacks a development regarding applications as this has been achieved for finite-state program verification. Especially more efficient algorithms for the construction of winning strategies would be useful, e.g. in control theory and "discrete event systems" ([RW89]).

We shall start with an introduction to the automata theoretic framework for studying infinite games. Here a game is specified by a "game graph" (a transition system in which the two players perform their transition steps) together with a "winning condition". We present a new proof of the Büchi-Landweber Theorem (as a construction of finite-state strategies in games over finite graphs with a winning condition of Muller type) and discuss some problems which arise for games over effectively presented infinite graphs.

I thank H. Lescow, S. Seibert and Th. Wilke, as well as the participants of the ASMICS Workshop "Transition systems with infinite behavior" (Bordeaux, November 1994), for many helpful discussions.

2 Definitions

2.1 State-based games

An abstract *infinite game* is given by an ω-language $\Gamma \subseteq A^\omega$ over an alphabet A (which is assumed to be finite in this paper). A *play* of the game is an ω-word $\alpha = a_0 a_1 a_2 \ldots$ over A, built up by two players 0 and 1 as follows: Player 0 picks a_0, player 1 picks a_1, player 0 picks a_2, and so on in alternation. The play α is won by 0 if $\alpha \in \Gamma$, otherwise the play is won by 1.

Often infinite games arise in a more concrete form than just by a set Γ of infinite words. A standard situation in computer science is the consideration of a transition system over a set of states, where actions induce steps from states to states, and plays are describable as state sequences. When ω-languages are specified by automata, this view is natural. It allows to model phenomena which are hidden in the abstract setting (assumed usually in descriptive set theory). For example, by means of states, periodicities in plays may be immediately captured by state repetitions, and also different potentials of performing actions (depending on the momentary state) are simply describable. In the following we introduce state-based games, using terminology from [Bü77], [Bü83], [GH82], [McN93], and [Ze94].

A *game graph* is of the form $G = (Q, Q_0, Q_1, A, \delta, \Omega)$ where Q is a finite or countable set of "states", Q_0, Q_1 define a partition of Q (Q_i containing the states where it is the turn of player i to perform an action), A is a finite set (of "actions"), and $\delta : Q \times A \rightharpoonup Q$ is a partial transition function. We require that the underlying graph is bipartite with respect to these transitions; formally, we have $\delta(Q_i \times A) \subseteq Q_{1-i}$ for $i = 0, 1$. Also for any $q \in Q$ some $a \in A$ is required where $\delta(q, a)$ is defined. Sometimes we designate a state q as "start state" and indicate the game graph by G_q.

The item Ω is an "acceptance component". In the sequel, we shall consider the cases where Ω is a state-set $F \subseteq Q$, a finite collection $\mathcal{F} = \{F_1, \ldots, F_m\}$ of state-sets, or a sequence $(E_1, F_1, \ldots, E_m, F_m)$ of sets of states from Q. We may view the states of the game graph "colored" correspondingly: a state q is colored by the 0-1-vector $c(q)$ of length 1, resp. m, resp. $2m$, where a 1 in the i-th component indicates that q belongs to the i-th set given in Ω. For a sequence $\gamma \in Q^\omega$ let $c(\gamma)$ be the sequence $c(\gamma(0))c(\gamma(1))\ldots$ of associated colors.

For a game graph G over Q, the decision who wins a play is fixed by a subset $\mathcal{C}$ of Q^ω, which we call *winning predicate*. We write "$\mathcal{C}(\gamma)$" if γ belongs to $\mathcal{C}$. If the acceptance component of G determines a coloring in $\{0,1\}^m$, we require that membership of γ in $\mathcal{C}$ is already fixed by the color sequence $c(\gamma) \in (\{0,1\}^m)^\omega$. The pair $(G, \mathcal{C})$ (or $(G_q, \mathcal{C})$) is a *state-based game*. A *play* in this game is a sequence $\gamma \in Q^\omega$ such that for any two succeeding states $\gamma(i), \gamma(i+1)$ there is an action a with $\delta(\gamma(i), a) = \gamma(i+1)$ (and such that $\gamma(0) = q$ if we deal with G_q). Player 0 *wins* the play γ if $\gamma \in \mathcal{C}$, otherwise player 1 wins.

A game $(G_q, \mathcal{C})$ may be considered as a (possibly infinite) ω-automaton, defining the ω-language which consists of all sequences $\alpha \in A^\omega$ which induce a play won by player 0.

A useful representation of a game $(G_q, \mathcal{C})$ is the unravelling of G_q in tree form, as *game tree* $t(G_q)$, which is again a game graph: Its states are the sequences $q_0 \ldots q_r$ which are possible initial segments of plays in G_q, and its transition function δ' is defined by $\delta'(q_0 \ldots q_r, a) = q_0 \ldots q_r \delta(q_r, a)$. The winning set $\mathcal{C}$ is adapted accordingly (referencing only q_r from a state $q_0 \ldots q_r$, but indicated again by $\mathcal{C}$). We call the game $(t(G_q), \mathcal{C})$ the *tree representation* of $(G_q, \mathcal{C})$.

2.2 Winning conditions

A *winning condition* is a formula (involving atomic formulas $c(\gamma(i)) = c_l$ for colors c_l) which defines a winning predicate $\mathcal{C}$. The most basic conditions refer to acceptance components of the form $\Omega = F$ (inducing a coloring in $\{0,1\}$):

- $\mathcal{C}(\gamma) :\equiv \exists i\ \gamma(i) \in F$ (formally: $\exists i\ c(\gamma(i)) = 1$) ($\Sigma^0_1$-*condition*),
- $\mathcal{C}(\gamma) :\equiv \forall i\ \gamma(i) \in F$ (Π^0_1-*condition*),
- $\mathcal{C}(\gamma) :\equiv \exists j \forall i \geq j\ \gamma(i) \in F$ (short: $\forall^\omega i\ \gamma(i) \in F$) ($\Sigma^0_2$-*condition*),
- $\mathcal{C}(\gamma) :\equiv \forall j \exists i > j\ \gamma(i) \in F$ (short: $\exists^\omega i\ \gamma(i) \in F$) ($\Pi^0_2$-*condition*).

For $k = 1, 2$, a game specified with a Σ^0_k-condition (Π^0_k-condition) is called a Σ^0_k-game (Π^0_k-game). More general winning conditions are first-order formulas with atomic formulas $R(i_1, \ldots, i_n)$ for any numerical relations R besides the atomic formulas $c(\gamma(i)) = c_l$; they define the finite-Borel games (where the winning predicate occurs on a finite level of the Borel hierarchy). The prefix of unbounded quantifiers in the prenex normal form of such a formula gives a bound for the level of the defined winning predicate in the Borel hierarchy.

An important class of games consists of the $\mathcal{B}(\Sigma_2^0)$-games, where the winning sets are defined by Boolean combinations of Σ_2^0-conditions. We use here two forms, namely, for acceptance components $\mathcal{F} = \{F_1, \ldots, F_m\}$

$$\mathcal{C}(\gamma) :\equiv \{f \mid \exists^\omega i \; \gamma(i) = f\} \in \mathcal{F} \quad (\textit{Muller condition}),$$

and for acceptance components $(E_1, F_1, \ldots, E_m, F_m)$

$$\mathcal{C}(\gamma) :\equiv \bigvee_{k=1}^{m} (\exists^{<\omega} \; \gamma(i) \in E_k \wedge \exists^\omega i \; \gamma(i) \in F_k) \quad (\textit{Rabin condition}).$$

These $\mathcal{B}(\Sigma_2^0)$-conditions are of special interest because they allow the specification of many properties which are relevant in concurrent systems (e.g. fairness properties, cf. [MP92]). Moreover, if games are considered as ω-automata, and games defining the same ω-language are regarded as equivalent, it is possible for finite-state games to reduce arbitrary S1S definable winning conditions to the Rabin condition and to the Muller condition (see, for example, [Th90]).

2.3 Strategies and Determinacy

A *strategy* for player 0, resp. 1, in the game $(G_q, \mathcal{C})$ as given above is a function σ which associates with any node $q_0 \ldots q_r$ of the game tree $t(G_q)$ (where $q_r \in Q_0$, resp. $q_r \in Q_1$) one of its successors in the game tree. A strategy σ thus determines a *fragment* of the game tree, obtained from the game tree by deleting all nodes ending in Q_1, resp. Q_0, which are not values of σ, together with their descendants. Let us denote this *strategy tree* by $t_\sigma(G_q)$. The function σ is called *winning strategy* for 0, resp. 1 in $(G_q, \mathcal{C})$ if for all paths γ through $t_\sigma(G_q)$ we have $\gamma \in \mathcal{C}$. (In this paper we do not consider *nondeterministic strategies*, as done in [GH82], [YY90], [Ze94].)

The existence of winning strategies is a central subject in descriptive set theory (see e.g. [Mos80]). The predominant question there is that of *determinacy*: For which games is it possible to guarantee that one of the two players has a winning strategy? If this holds the game is called *determined*. Determinacy is a powerful principle of complementation and hence important in mathematical logic: Nonexistence of a winning strategy for one player implies existence of a winning strategy for the other player. By a deep result of Martin [Ma75], a game is determined if its winning predicate is Borel. The games considered in this paper are all given with Borel winning predicates and hence determined.

In the questions of determinacy and complementation, the two players are handled symmetrically. For most applications in distributed systems, however, one identifies the players with two parties which are handled asymmetrically: one of them represents the program (or "control", here identified with player 0), the other represents the environment (or "disturbance", here identified with player 1). Then it is more interesting that player 0 has a winning strategy than just one of the two players. For example, an operating system (player 0) should function in an arbitrary or even hostile environment (player 1). Determinacy is helpful or necessary, however, in proofs that a strategy construction is complete, or when an induction refers to "smaller" games where a strategy can be guaranteed just for one of the players (and not for a fixed one).

We are mainly interested in the effective presentation and the easy execution of strategies for a given player, assuming that the games are presented as finite objects. For instance, we consider recursive games. (Such a game would be specified by an algorithm which allows to compute predecessors and successors of the nodes in the game tree, their colors, and their association with players 0 and 1.) For effectively presented games (of a given type), the following questions arise:

1. Does player 0 have a winning strategy?
2. Is there an effective (or efficient) construction of winning strategies for player 0 from game presentations?
3. How to construct winning strategies of low computational complexity?

The second question is concerned with the *construction* of winning strategies, the third one with the *execution* of strategies. In the former case an effective or efficient transformation from representations of games to representations of strategies is required, while in the latter case efficient algorithms to compute values of strategies (i.e., values of word functions σ over the state space) are asked for.

In the simplest case, where a value $\sigma(q_0 \dots q_r)$ of a strategy only depends on the last state q_r, we speak of a *no-memory strategy.* Such strategies are representable as fragments of the game graphs: For example, a no-memory strategy for player 0 is specified by keeping only a single edge from any state of Q_0 in a game graph over Q. If a strategy σ is computable by a finite automaton (say by a Moore automaton), it is called a finite-memory strategy or *finite-state strategy.* Another important type of strategy is that of *recursive* (effectively executable) strategies.

3 Finite-state games

In this section we outline an "incremental" proof of the Büchi-Landweber Theorem of [BL69]: Given a finite-state game with Muller winning condition, the partition of the state space into the sets of states from which player 0, resp. player 1, wins is computable, and corresponding winning strategies are effectively constructible as finite-state strategies. We proceed in four steps, covering winning conditions of inceasing difficulty (Σ_1^0- and Π_2^0-condition, a special type of Rabin condition, and the Muller condition).

3.1 Σ_1^0- and Π_2^0-games

The first step deals with Σ_1^0-games. For a game graph $G = (Q, Q_0, Q_1, A, \delta, F)$, we have to determine the set of those states from which player 0 can force a visit of some state in F. This is done by computing, for $i \geq 0$, the sets W_i of states from which a visit in F can be forced in at most i steps. Clearly we have $W_0 = F$ and

$$\begin{aligned} W_{i+1} = W_i &\cup \{p \in Q_0 \mid \exists a \in A \ \delta(p,a) \in W_i\} \\ &\cup \{p \in Q_1 \mid \forall a \in A \ \delta(p,a) \in W_i \text{ if defined}\}. \end{aligned}$$

Since the sequence (W_i) is increasing and Q is finite, there is some k where this sequence becomes constant and is the union of the W_i. Let Reach(F) be W_k for the first such k. It is easy to see this set contains the states from which 0 wins the game. The winning strategy is best described by the ranking of states as determined by the sets W_i (the rank of $p \in \text{Reach}(F)$ being the smallest i such that $p \in W_i$). Now the strategy just has to ensure that the rank decreases with each step, which can be done by deleting all edges violating this condition. We obtain a no-memory strategy. A no-memory strategy for the opposite player 1 applies to any state in $Q_1 \setminus \text{Reach}(F)$, specifying a transition to a target state again outside Reach(F) (which exists by definition of Reach(F)).

In the second step we treat Π^0_2-games, again given by game graphs of the form $G = (Q, Q_0, Q_1, A, \delta, F)$. We determine the states from which infinitely many visits of F can be forced by player 0 (and skip here the proof that player 1 wins from the remaining states). For this, we first compute the states from F (forming a set Recur(F)) which allow player 0 to force infinitely many revisits of F. In an auxiliary step we define, for any given state set V, the set of states from which a single revisit in V can be forced in ≥ 1 steps. For this purpose, one modifies the definition of the sets W_i above. Let

$$\begin{aligned} W'_1 = \{p \in Q_0 \mid \exists a \in A \ \delta(p,a) \in V\} \\ \cup \{p \in Q_1 \mid \forall a \in A \ \delta(p,a) \in V \text{ if defined}\} \end{aligned}$$

and obtain W'_{i+1} from W'_i as W_{i+1} from W_i above. Let us denote by $\text{Reach}^+(V)$ the first set W'_k such that $W_k = W'_{k+1}$; precisely from its states a visit of V can be forced by player 0 in ≥ 1 steps. Now let V_i be the set of states from which player 0 can force at least i visits to F. We have

$$V_1 = F, \quad V_{i+1} = V_i \cap \text{Reach}^+(V_i)$$

The sequence (V_i) is decreasing. Let Recur(F) be their intersection, i.e., V_l for the smallest l with $V_l = V_{l+1}$. In the considered Π^0_2-game, player 0 has now a winning strategy from those states which are in Reach(Recur(F)). Again it is a no-memory strategy by the inductive definition of the sets W'_i (where one imposes to stay within V_l when it is entered).

3.2 $\mathcal{B}(\Sigma^0_2)$-games

We discuss $\mathcal{B}(\Sigma^0_2)$-games in two stages. The first refers to the so-called *Rabin chain condition* ([Mst84]), the second to the Muller condition. A Rabin chain condition over a state set Q is given by a sequence $(E_1, F_1, \ldots, E_m, F_m)$ with the extra property that $E_1 \subset F_1 \subset \ldots \subset E_m \subset F_m$ $(\subseteq Q)$. As in Section 2.2, the winning condition requires for a play $\gamma \in Q^\omega$ that for some k, infinitely visits in F_k occur but only finitely many visits in E_k (finally excluding E_k).

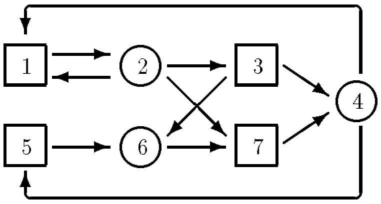
The construction of no-memory strategies for games with such winning conditions works by induction on the size of the state space. We follow a construction of McNaughton ([McN93, Thm. 6.2]), given there for winning conditions of a related form (Muller conditions "without splits"). The claim is that any game graph with Rabin chain winning condition allows a partition into two sets W_0, W_1 such that player i has a no-memory winning strategy from the states in W_i.

Consider the game graph $G = (Q, Q_0, Q_1, A, \delta, \Omega)$ with acceptance component $\Omega = (E_1, F_1, \ldots, E_m, F_m)$. If $|Q| \leq 2$ the claim is trivial. In the induction step, assume that $E_1 = \emptyset$, i.e., F_1 is the first nonempty entry in Ω (otherwise switch players 0 and 1). Pick $q \in F_1$. Then infinitely many visits of q suffice to ensure a win of player 0. (Note that, by the chain condition and minimality of F_1, there is no way to extend $\{q\}$ to a set of infinitely often visited states that causes a win of player 1.) One verifies that $Q \setminus \mathrm{Reach}(\{q\})$ is again a game graph G_0; thus the induction hypothesis yields a partition of this game graph into sets V_0, V_1 such that player i has a no-memory winning strategy in G_0 playing from states in V_i.

We now distinguish the following two (complementary!) cases: Either player 0 can force a direct transition from q to $Q \setminus V_1$, or player 1 can force a direct transition to V_1. We claim that in the first case, the states from which 0 wins are those in $Q \setminus V_1$ (which is the set $\mathrm{Reach}(\{q\}) \cup V_0$): Player 0 works from any state in $\mathrm{Reach}(\{q\})$ towards q and (by case 1) repeats this or proceeds to V_0 where the existing winning strategy is applied as long as player 1 allows to stay in V_0. If 1 chooses to leave V_0, then necessarily by a transition into $\mathrm{Reach}(\{q\})$ which means 0 can force a visit to q again. (Transitions from V_0 to V_1 by player 1 are impossible by the induction hypothesis on G_0.) Thus, player 1 only has the options to remain in V_0 from some moment onwards or to pass through q again and again. In either option, player 0 wins (using an extension of the given no-memory strategy on V_0 to a no-memory strategy also on $\mathrm{Reach}(\{q\})$). The required winning strategy for 1 is the one given by the induction hypothesis.

It remains to treat the case that player 1 can force a direct transition from q into V_1. Consider all states (including q, of course) from which player 1 can force a visit in V_1 over the game graph G. Call this set V; on V there is a no-memory winning strategy for 1. The complement of V in G turns out to be a game graph to which (by absence of q) again the induction hypothesis can be applied. Its decomposition into two sets U_0, U_1 yields the desired partition of G, namely U_0 as the set of states from which 0 wins, and $V \cup U_1$ as the set of states from which 1 wins, both by a no-memory strategy.

Finally, we consider $\mathcal{B}(\Sigma_2^0)$-games presented with the Muller winning condition. Here a collection $\mathcal{F} = \{F_1, \ldots, F_m\}$ of state sets is given, and the winning condition for player 0 requires that the states visited infinitely often in a play form one of the sets F_k. (Thus, only loops, i.e. strongly connected state sets, are reasonable candidates of such sets F_k.) Let us verify that winning strategies without memory do not suffice, by considering the example game graph displayed in the figure. The even-numbered circles represent the states in Q_0, the odd-numbered boxes those of Q_1. We suppress the labels on edges (actions).



Since all possible loops are uniquely determined by their odd-numbered states, we name the final state sets just by their odd-numbered elements. Suppose the loops $\{1, 3\}$ and $\{1, 3, 5, 7\}$ are those in which player 0 wins (defining the set $\mathcal{F}$). Then a no-memory strategy would have to select one edge at state 4, but any of the two possible choices would enable player 1 to win (obviously for the choice of 5, and for the choice of state 1 by passing from 3 to 6 when state 3 is reached).

A little contemplation will show that at state 4 it should be known how it was reached: If 4 was reached directly from 3, then a good choice is to go to state 1 and from 2 always to 3, which means that upon repetition of this process the loop $\{1, 3\}$ is assumed forever and player 0 wins. If, however, state 4 is reached from 3 via 6 and 7, then it is advisable to force a visit of all odd-numbered states: first 5 and again 7, then 1 and 3. This can be executed, for example, by a memory which always records the last two visits to odd-numbered states. (Assuming that player 0 moves from 2 always to 3, there are three possibilities of last visits at state 4: $(1, 3)$, $(5, 7)$, and $(3, 7)$. The first two cases require a transition to state 1, the last case a transition to 5.) Altogether the strategy is implementable by a Moore automaton and hence finite-state.

The record on last visits of states is a basic tool for building strategies: it is found under the name *later appearance record* LAR in [GH82], *order-vector* in [Bü83], and *latest visitation record* in [McN93]. For a general construction, we shall use the LAR here in an extended form following [Bü83]: Over the state-set $Q = \{1, \ldots, n\}$, an LAR with *hit position* is a permutation $(i_1, \ldots, i_n)$ of $(1, \ldots, n)$ together with a number h from $\{1, \ldots, n\}$ (the "hit"). We write this as an n-tuple where position h is underlined and identify the set of all these extended vectors with $\mathrm{Perm}(Q) \times Q$. Formally, we introduce the extended LAR of a finite state-sequence inductively: For the empty sequence ε over $Q = \{1, \ldots, n\}$, $\mathrm{LAR}(\varepsilon)$ is $(1, \ldots, \underline{n})$. For a state sequence $s.q$, $\mathrm{LAR}(s.q)$ is obtained by taking q from its position j in $\mathrm{LAR}(s)$ towards the end, and setting the hit to be j. So the hit records from which place onwards in an LAR a change has just occurred. In our example above (disregarding again all even-numbered states), a repeated tour through the loop $\{1, 3\}$ will lead to LAR-values $(5, 7, \underline{1}, 3)$ and $(5, 7, \underline{3}, 1)$, whereas the repeated tour through all states will set the hit to the first position again and again. Using the hit position, a set F can be fixed as containing precisely the states visited infinitely often, in the following way (assuming $|F| = k$): From some moment onwards, the hit stays $\geq (n-k)+1$, and infinitely often the hit is $(n-k)+1$ with the elements of F (in some order) on the LAR positions $(n-k)+1, \ldots, n$. Thus, the hit induces a scale for the final sets by size (which supplies the connection with the Rabin chain condition).

Given a game graph $G = (Q, Q_0, Q_1, A, \delta, \mathcal{F})$, we shall extract the desired finite-state winning strategies for players 0 and 1 from a new game graph $G' = (Q', Q'_0, Q'_1, A, \delta', \Omega)$, which is equipped with a winning condition in Rabin chain form. We set $Q' = Q \times \mathrm{Perm}(Q) \times Q$, and let a state of Q' be in Q'_0 (resp. Q'_1) iff its first component is in Q_0 (resp. in Q_1). The transition function δ' copies δ regarding the first component, while for the remaining components δ' realizes the update of the LAR as explained above. A play γ over G thus corresponds to a play γ' over G', using the initial LAR for the first state $\gamma'(0)$. We shall define the acceptance component Ω in Rabin chain form such that the following equivalence holds:

> A play γ over G is won by player 0 (resp. player 1) iff the corresponding play γ' over G' is won by player 0 (resp. player 1).

Using this, we can apply the previous strategy construction to G' (since over G' the winning condition is in Rabin chain form) and obtain a no-memory strategy for any $q' \in Q'$ where there is a winning strategy for player 0 (resp. player 1) in $G'_{q'}$ at all. The strategy is obtained by deleting certain transitions in G', which in turn defines the desired finite-state strategy over G (since the storage and updates of the extended LAR can be realized by a finite automaton).

It remains to establish the above equivalence. For this, we assume $|Q| = n$ and define a chain $\Omega = (E_1, F_1, \ldots, E_n, F_n)$ of subsets of Q'. Let E_j contain all states from Q' where the LAR part has a hit value smaller than j, and let F_j be the union of E_j with the set of states from Q' where this hit value is precisely j and the LAR-entries from the hit position onwards form a set in $\mathcal{F}$. (We obtain a proper inclusion chain by merging E_j and E_{j+1} if $F_j \setminus E_j = \emptyset$, resp. F_j and F_{j+1} if $E_{j+1} \setminus F_j = \emptyset$.) It is now easy to verify the claimed equivalence, which completes the strategy construction for finite-state $\mathcal{B}(\Sigma^0_2)$-games.

In our example above, we could work just with the set of odd-numbered states in place of Q. In [McN93], such a set W of "relevant states" is introduced in general: It contains just enough states for the distinction between final and non-final loops. We see from the proof above that in a game over Q with a Muller condition over a set W of relevant states, the winner has a winning strategy requiring a memory of size $|W|! \cdot |W|$. A recent (and quite different) proof of S. Seibert ([Se94]) gives a proper exponential upper bound of $4^{|W|}$ for the memory. Examples of H. Lescow and S. Seibert show that for some constant c, $c^{|W|}$ is a lower bound for the memory size of the winner. It would be interesting to know classes of games (or winning conditions) where the winner has a polynomial size (however non-zero memory) winning strategy.

The proof above shows that the Rabin chain condition is a useful intermediate step in handling games (or automata) with the Muller condition. We see how the game presentations are related to the memory-size in winning strategies: When game graphs are considered as ω-automata, the transformation of automata with

Muller acceptance to those with Rabin chain acceptance involves a blow-up in the state space which supplies exactly the memory suitable to win games with Muller winning condition as compared to the (zero) memory for games with Rabin chain condition.

The Rabin chain condition was introduced by Mostowski in [Mst84] but remained rather unnoticed. It was applied independently for obtaining no-memory strategies in [Mst91a] and [EJ91] (where it is called "parity condition"). The simulation of Muller acceptance using Büchi's version of the LAR, as presented here, yields smaller transition graphs with Rabin chain condition than previous constructions in [Mst91b] and [Ca94].

Gurevich and Harrington showed the much more general result that strategies with an LAR memory suffice for $\mathcal{B}(\Sigma_2^0)$-games even over infinite graphs ("Forgetful Determinacy Theorem" [GH82], [YY90], [Ze94]). By lack of space we cannot enter this interesting subject here. For example, fixed point constructions (as above in Section 3.1 for the finite-state case) require now ordinal-indexed sets. Games of this form can be applied to obtain the complementation of Rabin tree automata (as proposed by Büchi [Bü77], [Bü83]). Several approaches have been developed to obtain transparent proofs; as recent references, besides the papers listed above, we mention [EJ91], [Mu92], [Kl94], [MS94].

4 Games over pushdown transition graphs and recursive graphs

The subject of this section is still in its beginnings: effective winning strategies (and their construction) for games on effectively presented infinite graphs.

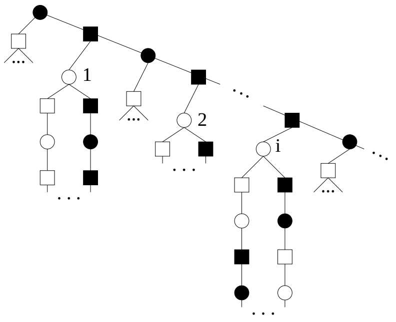
A natural first step after finite-state games is to consider games over transition graphs of deterministic pushdown automata. Here each node corresponds to a global state of a pushdown automaton, given by the content of the pushdown store and the state of the finite control. These are *context-free graphs* in the sense of Muller and Schupp ([MS85]), also called "context-free processes" in semantics of concurrency, and their unravellings in tree form are the *algebraic trees* in the sense of [Co83]. As Courcelle has shown in [Co94], the monadic second-order theory of an algebraic tree is decidable. This result can be applied to games over algebraic trees where the winning conditions are expressible in monadic second-order logic. In this case, the existence of a winning strategy is expressed by a monadic second-order sentence about the game tree, saying "there is a fragment of the game tree defining a strategy, such that for each path through the fragment, the winning condition is satisfied". (Analogously, the Büchi-Landweber-Theorem can be shown using finite tree automata over infinite trees and the effective solution of their emptiness problem, cf. [Ra72].) One should expect that from the decidability proof of [Co94] also an effective construction of a winning strategy can be extracted (if such a strategy exists); however, this sharpened claim and a complexity analysis are presently open.

In contrast to the finite-state case, there are now natural winning conditions which are no more monadic second-order definable and occur at higher Borel levels than $\mathcal{B}(\Sigma_2^0)$. An example is the winning condition on paths through a pushdown transition graph which requires that some pushdown content (possibly with an extra property) occurs infinitely often. The condition is in Σ_3^0-form, and it seems interesting to decide the existence of winning strategies for a given player, to analyze if and when these strategies are recursive, and to effectively. So far, only Büchi's work [Bü8

${}_3^0 \cap \Pi_3^0$)-games

seems to be available on this subject. (In such games, it can also be appropriate to admit infinitely many colors, e.g. representing the infinitely many pushdown contents, in order to reformulate the winning condition.)

In *recursive games*, the construction of effective strategies may fail even for Π_2^0-winning conditions. To verify this, consider the following game tree t:



The tree is defined using the halting problem for Turing machines in the following way: Below the node named i in the figure, we have a switch of colors (as indicated) on the k-th level of the two branches iff the i-th Turing machine M_i halts on the empty tape after k steps. Clearly the tree t is recursive. Consider the game $(t, \mathcal{C})$ with winning condition "$\mathcal{C}(\gamma) :\equiv \exists^\omega n\ \gamma(n)$ is colored black". It is obvious that a winning strategy for 0 in this game is recursive iff the halting problem for Turing machines is recursive (because at node i a strategy has to fix a decision which is equivalent to the decision whether M_i eventually halts).

A more drastic statement can be obtained from the following recursion theoretic result: There is a nonempty Π_2^0-ω-language $L \subseteq \{0,1\}^\omega$ such that no ω-word $\gamma \in L$ is hyperarithmetical, i.e., for any $\gamma \in L$ the numbers i with $\gamma(i) = 1$ form a set of natural numbers outside Δ_1^1. (See e.g. [Mos80, 4.D.10], using that in the Cantor space $\{0,1\}^\omega$, Σ_1^1-sets are projections of Π_2^0-sets.) Such an ω-language L contains the ω-words $\alpha \in \{0,1\}^\omega$ such that $\exists^\omega i\ \alpha(0)\ldots\alpha(i) \in R$ for a certain recursive set R; L induces a full binary recursive game tree t_L where the root is associated with player 0 and a node $a_1 a_2 \ldots a_{2i} a_{2i+1} \in \{0,1\}^+$ is colored "black" iff $a_1 a_3 \ldots a_{2i+1} \in R$. A strategy for 0 defines a path γ through t_L and is of the same recursion theoretic degree as γ. Using again the winning condition "$\mathcal{C}(\gamma) :\equiv \exists^\omega n\ \gamma(n)$ is black", there is a winning strategy for 0 (because $L \neq \emptyset$), but none which is hyperarithmetical.

As above for context-free games, special but natural classes of recursive games should be found such that there are recursive winning strategies, which moreover should be obtained effectively from the game presentations.

5 Conclusion

We conclude with a remark on two general research directions which were not touched in this paper.

For applications in control systems, it is important to connect the discrete model considered above with the possibility of continuous changes of parameters in time. References on games over such "hybrid systems" are [NYY92b] and [MPS94].

Finally, we mention a somewhat vague problem: Is there a logical framework for the specification of games such that winning strategies can be built up in a "compositional" manner? The present automata theoretic methodology uses two nontrivial and often impractical transformations when starting from logical specifications: First the conversion of logical formulas to automata (and game graphs), which is costly when the specifications involve many quantifier alternations, secondly the conversion of game graphs to strategies, which can be costly for games with $\mathcal{B}(\Sigma_2^0)$-winning conditions. A logic of system specification which would allow to construct strategies more directly (at least in some interesting cases) would be both of theoretical and practical value.

References

[ALW89] M. Abadi, L. Lamport, P. Wolper, Realizable and unrealizable specifications of reactive systems, in: *Proc. 17th ICALP* (G. Ausiello et al., eds.), Lecture Notes in Computer Science **372** (1989), Springer-Verlag, Berlin 1989, pp. 1-17.

[Bü62] J.R. Büchi, On a decision method in restricted second order arithmetic, in: *Proc. Int. Congr. Logic, Method. and Philos. of Science* (E. Nagel et al., eds.), Stanford Univ. Press, Stanford 1962, pp. 1-11.

[Bü77] J.R. Büchi, Using determinacy to eliminate quantifiers, in: *Fundamentals of Computation Theory* (M. Karpinski, ed.), Lecture Notes in Computer Science 56, Springer-Verlag, Berlin 1977, pp. 367-378.

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

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

[Ca94] O. Carton, Chain automata, in: *Technology and Applications,* Information Processing '94, Vol. I (B. Pherson, I. Simon, eds.), IFIP, North-Holland, Amsterdam 1994, pp. 451-458.

[Ch63] A. Church, Logic, arithmetic and automata, *Proc. Intern. Congr. Math. 1962*, Almqvist and Wiksells, Uppsala 1963, pp. 21-35.

[CGL94] E. Clarke, O. Grumberg, D. Long, Verification tools for finite-state concurrent systems, in: *A Decade of Concurrency* (J.W. de Bakker et al., eds.), Lecture Notes in Computer Science **803**, Springer-Verlag, Berlin 1994, pp. 124-175.

[Co83] B. Courcelle, Fundamental properties of infinite trees, *Theor. Comput. Sci.* **25** (1983), 95-169.

[Co94] B. Courcelle, The monadic second-order theory of graphs IX: Machines and their behaviours, Techn. Report, LaBRI; Université Bordeaux I, 1994.

[EJ91] E.A. Emerson, C.S. Jutla, Tree automata, Mu-calculus and determinacy, in: *Proc. 32th Symp. on Foundations of Computer Science* (1991), 368-377.

[GH82] Y. Gurevich, L. Harrington, Trees, automata, and games, in: *Proc. 14th ACM Symp. on the Theory of Computing*, San Francisco, 1982, pp. 60-65.

[Kl94] N. Klarlund, Progress measures, immediate determinacy, and a subset construction for tree automata, *Ann. Pure Appl. Logic* **69** (1994), 243-168.

[Ma75] D. Martin, Borel determinacy, *Ann. Math.* **102** (1975), 363-371.

[McN93] R. McNaughton, Infinite games played on finite graphs, *Ann. Pure Appl. Logic* **65** (1993), 149-184.

[MP92] Z. Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Programs*, Springer-Verlag, Berin, Heidelberg, New York 1992.

[Mos80] Y. N. Moschovakis, *Descriptive Set Theory*, North-Holland, Amsterdam 1980.

[MPS94] O. Maler, A. Pnueli, J. Sifakis, On the synthesis of discrete controllers for timed systems, these Proceedings.

[Mst84] A.W. Mostowski, Regular expressions for infinite trees and a standard form of automata, in: A. Skowron (ed.), *Computation Theory*, Lecture Notes in Computer Science **208**, Springer-Verlag, Berlin 1984, pp. 157-168.

[Mst91a] A.W. Mostowski, Games with forbidden positions, Preprint No. 78, Uniwersytet Gdański, Instytyt Matematyki, 1991.

[Mst91b] A.W. Mostowski, Hierarchies of weak automata and weak monadic formulas, *Theor. Comput. Sci.* **83** (1991), 323-335.

[Mu92] A. Muchnik, Games on infinite trees and automata with dead-ends. A new proof for the decidability of the monadic second-order theory of two successors, *Bull. of the EATCS* **48** (1992), 220-267.

[MS85] D.E. Muller, P.E. Schupp, The theory of ends, pushdown automata, and second-order logic, *Theor. Comput. Sci.* **37** (1985), 51-75.

[MS94] D.E. Muller, P.E. Schupp, Simulating alternating tree automata by nondeterministic automata: new results and new proofs of the theorems of Rabin, McNaughton and Safra, *Theor. Comput. Sci.* (to appear).

[NYY92a] A. Nerode, A. Yakhnis, V. Yakhnis, Concurrent programs as strategies in games, in: *Logic from Computer Science* (Y. Moschovakis, ed.), Springer, 1992.

[NYY92b] A. Nerode, A. Yakhnis, V. Yakhnis, Modelling hybrid systems as games, in: *Proc. 31st IEEE Conf. on Decision and Control*, Tucson, pp. 2947-2952.

[PR89] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: *Proc. 16th ACM Symp. on Principles of Progr. Lang.*, Austin, pp. 179-190.

[Ra72] M.O. Rabin, Automata on infinite objects and Church's Problem, Amer. Math. Soc., Providence, RI, 1972.

[RW89] P.J.G. Ramadge, W.M. Wonham, The control of discrete event systems, *Proc. of the IEEE* **77** (1989), 81-98.

[Se94] S. Seibert, Doctoral Thesis, in preparation.

[Th90] W. Thomas, Automata on infinite objects, in: J. v. Leeuwen (ed.), Handbook of Theoretical Computer Science, Vol. B., Elsevier Science Publ., Amsterdam 1990, pp. 133-191.

[YY90] A. Yakhnis, V. Yakhnis, Extension of Gurevich-Harrington's restricted memory determinacy theorem: A criterion for the winning player and an explicit class of winning strategies, *Ann. Pure Appl. Logic* **48** (1990), 277-297.

[Ze94] S. Zeitman, Unforgettable forgetful determinacy, *J. Logic Computation* **4** (1994), 273-283.

This article was processed using the LaTeX macro package with LLNCS style