Church's Problem and a Tour through Automata Theory

Wolfgang Thomas

RWTH Aachen, Lehrstuhl Informatik 7, 52056 Aachen, Germany
thomas@informatik.rwth-aachen.de

Dedicated to Boris A. Trakhtenbrot, pioneer and teacher of automata theory for generations of researchers, on the occasion of his 85th birthday.

Abstract. Church's Problem, stated fifty years ago, asks for a finite-state machine that realizes the transformation of an infinite sequence α into an infinite sequence β such that a requirement on (α, β), expressed in monadic second-order logic, is satisfied. We explain how three fundamental techniques of automata theory play together in a solution of Church's Problem: Determinization (starting from the subset construction), appearance records (for stratifying acceptance conditions), and reachability analysis (for the solution of games).

1 Introduction

Around 1960, a core of automata theory had been established which led to the first comprehensive expositions, such as the volume *Sequential Machines – Selected Papers* edited by Moore [15] and the monograph [11] of Hopcroft and Ullman. In these early books three essential aspects of automata theory are either underrepresented or missing: the view of automata as transducers (computing functions rather than accepting languages), the use of automata in the study of infinite computations, and the close connection between automata and logic. These directions of study are a focus in the work of B.A. Trakhtenbrot. In the development of automata theory the three aspects often appeared in combination, offered most beautiful results and – as we know today – are highly significant and even indispensable for many applications in the design and analysis of computer systems.

The breakthrough on the relation between automata and logic was the proof of the expressive equivalence between finite automata and weak monadic second-order arithmetic over the natural number ordering, established by Büchi and Elgot (see the joint announcement [3] of 1958 and the two papers [1,8]) and independently by Trakhtenbrot [27] (submitted in July 1957). The Büchi-Elgot-Trakhtenbrot Theorem was extended soon after by Büchi [2] to the full monadic second-order theory of the natural number ordering, together with an expressive model of finite automaton over infinite sequences ("Büchi automaton"). Later Rabin showed how to generalize this theory to cover also infinite trees [20].

A. Avron et al. (Eds.): Trakhtenbrot/Festschrift, LNCS 4800, pp. 635–655, 2008.

The equivalence between formulas of monadic second-order logic and finite automata opened a way to establish algorithms that can test sentences for truth in the standard model of arithmetic. After decades of work on variants of the original question and on improving the efficiency of decision procedures, this approach became the origin of "model-checking", today a vast field which offers techniques for verifying highly nontrivial software and hardware systems.

Regarding the infinite behavior of automata and the use of automata as transducers, a master problem was raised by Church in 1957 [5] (see also [6]). He asked for the synthesis of automata that realize functions over infinite words rather than languages. Church posed the problem whether certain transformations of infinite words that are specified in a system of arithmetic are computable by finite automata (in his words: by circuits):

> Given a requirement which a circuit is to satisfy, we may suppose the requirement expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The *synthesis problem* is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit). ([5, p.8-9])

Church's Problem was solved by Büchi and Landweber [4] for specifications in monadic second-order logic over $(\mathbb{N}, <)$, building on a fundamental result by McNaughton [13] on the determinization of Büchi automata. These two results, the Büchi-Landweber Theorem and the McNaughton Theorem, are the origin of a field which might be called "synthesis of reactive systems" (rather than "verification"), with the algorithmic theory of infinite games as a core discipline. Today the area attracts much attention – it is concerned with refined studies on Church's synthesis problem and the extension to more general questions (e.g., on infinite stochastic games or multiplayer games).

At a very early stage, it was again Trakhtenbrot who merged the fundamental constructions of the subject in his pioneering monograph with Barzdin [28]. The part due to Trakhtenbrot (namely, Chapters I to III) covers both key results indicated above; it is based on lecture notes of his of 1966, with more material (on the Büchi-Landweber Theorem) added with the translation. It is remarkable to see that the authors call (in the preface to [28]) the Chapters I-III the "old" parts of the theory, while just the Chapters IV and V, which focus on statistical aspects, are mentioned as the "first encouraging steps of a new trend". This judgment was prophetic in the sense that today it is true as it was more than 30 years ago; probably the use of statistical methods will be a key in developing efficient approaches to the present demanding challenges in verification and synthesis. However, for the remainder of this paper, our objective is rather to reflect on the "old" theory. We single out basic ingredients that are relevant to Church's Problem, taking a view as it developed over the past twenty years. We call these methods "determinization", "appearance records", and "reachability analysis". The first two deal with approaches to set up memory structures in finite automata, and the last one is concerned with techniques for exploring transition graphs. The purpose of this paper is to present the integration of these ideas in

a solution of Church's problem. Since the details of these constructions are well-known, our exposition focusses on methodological issues rather than offering a full technical treatment[1].

In Sect. 2, we begin with a presentation of Church's Problem. Section 3 briefly discusses the issue of determinization and subset constructions. Determinization is the key construction for transforming Church's Problem into a problem of state-based infinite games, namely into the question of solving so-called "Muller games".

Sections 3 and 4 are the main part of the paper; they are devoted to the solution of Muller games in two stages, following an idea proposed in [24] (as an alternative to the original proof in [4]). The first stage is a stratification of the Muller winning condition; it leads to the so-called "parity condition". We explain how this stratification is obtained by a simple memory structure that we call "appearance records". We present it in two versions (for weak and strong Muller games).

The last step is the solution of parity games (again in their weak and strong version), showing memoryless determinacy of these games. This completes the solution of Church's Problem. We explain (in Sect. 4) that the core of the proof of memoryless determinacy of parity games is provided by (a subtle iteration of) simple reachability tests. In the present game theoretical framework, we deal with alternating reachability.

Of course, this emphasis on three essential constructions just points to some selected central ideas. Automata theory is much too rich to be reducible to these simple principles. For example, we do not touch the large area of automaton minimization. Our exposition is also more motivated by didactic aspects than by claims on practical applicability. For applications in program verification or program synthesis, one often has to find refinements or even alternatives for the basic constructions in order to ensure algorithmically satisfactory solutions.

2 Church's Problem

Let us start with an example. Our objective is to construct a finite automaton that transforms an input stream α of bits into an output stream β of bits such that the following three conditions are satisfied. (We write, e.g., $\alpha(t)$ for the t-th bit of α $(t = 0, 1, \ldots)$, and $\exists^\omega$ for the quantifier "there exist infinitely many".)

1. $\forall t(\alpha(t) = 1 \quad \rightarrow \quad \beta(t) = 1)$
2. $\neg\exists t\ \beta(t) = \beta(t+1) = 0$
3. $\exists^\omega t\ \alpha(t) = 0 \quad \rightarrow \quad \exists^\omega t\ \beta(t) = 0$

The desired automaton has to produce the output bit $\beta(t)$ without delay upon receipt of $\alpha(t)$. More specifically, we work with transducers in the format of deterministic Mealy automata. A Mealy automaton has the format $\mathcal{M} = (S, \Sigma, \Gamma, s_0, \delta, \tau)$ where S is the finite set of states, Σ and Γ are the input

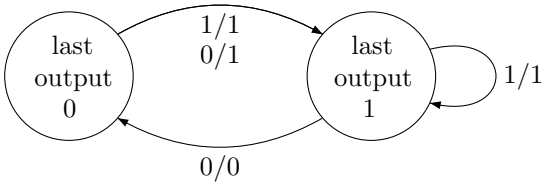
[1] Readers who want to see a self-contained exposition are referred to the tutorial [25].

alphabet and output alphabet, respectively, s_0 the initial state, $\delta : S \times \Sigma \to S$ the transition function and $\tau : S \times \Sigma \to \Gamma$ the output function. In a graphical presentation we label a transition from p to $\delta(p, a)$ by $a/\tau(p, a)$. The definition of the function $f_{\mathcal{M}} : \Sigma^\omega \to \Gamma^\omega$ computed by $\mathcal{M}$ is then obvious.

For our example, the first two conditions are satisfied easily by producing output 1 at each moment t. But the last condition, which has the form of a fairness constraint, excludes this simple solution; we cannot ignore the zero bits in α. A natural idea is to alternate between outputs 0 and 1 if the inputs are only 0. We arrive at the following procedure:

- for input 1 produce output 1
- for input 0 produce
  - output 1 if last output was 0
  - output 0 if last output was 1

This procedure is executable by the following Mealy automaton. (As initial state we take, for example, the left-hand state.)



Let us present the specification language and the task of synthesis in more detail. For the formulation of "requirements" we consider the system of monadic second-order logic (MSO) over the successor structure $(\mathbb{N}, +1, <)$, also called S1S (for "second-order theory of one successor") or "sequential calculus". This case was emphasized by Church as an open problem in [6], and today it is understood that "Church's Problem" refers to S1S. In short words, this language uses variables $s, t, \ldots$ for time instances (natural numbers) and variables $X, Y, \ldots$ for sequences. Sequences are identified here with unary predicates over the natural numbers: The bit sequence α is identified with the predicate that holds for t iff $\alpha(t) = 1$; so in S1S one writes $X(t)$ rather than $\alpha(t) = 1$. The atomic formulas are equalities and inequalities between number terms (e.g. $s + 1 + 1 = t$, $s < t$) and formulas $X(\tau)$ with number term τ; the S1S-formulas are built from atomic ones by applying Boolean connectives and the quantifiers $\forall, \exists$ to both kinds of variables. In general, we have S1S-specifications $\varphi(\overline{X}, \overline{Y})$ that speak about *tuples* of predicates (sequences). For an m_1-tuple $\overline{X}$ and an m_2-tuple $\overline{Y}$ this means that the input alphabet under consideration is $\{0,1\}^{m_1}$ and the output alphabet $\{0,1\}^{m_2}$. In our explanations and examples we only refer to the case $m_1 = m_2 = 1$.

Church's Problem can now be stated as follows:

> *Given an S1S-specification $\varphi(\overline{X}, \overline{Y})$, decide whether a Mealy automaton exists that transforms each input sequence $\alpha \in (\{0,1\}^{m_1})^\omega$ into an output sequence $\beta \in (\{0,1\}^{m_2})^\omega$ such that $(\mathbb{N}, +1, <) \models \varphi[\alpha, \beta]$ – and if this is the case, construct such an automaton.*

Among the many concepts of transformations of sequences, only a very special form is admitted for Church's Problem. Two aspects are relevant, as was at an early stage clarified by Trakhtenbrot [26]: First, the transformation should be "causal" (or: "nonanticipatory"), which means that the output $\beta(t)$ only depends on the prefix $\alpha(0)\dots\alpha(t)$ of α. (Thus we have a much sharper requirement than continuity in the Cantor space, where $\beta(t)$ is determined by some finite prefix of α, possibly longer than $\alpha(0)\dots\alpha(t)$.) The second aspect is the computability of the transformation by a finite-state machine (and here we take the above-mentioned format of Mealy automata, to be specific).

As an illustration consider the two transformations T^- and T^+ which "divide by 2", respectively "double" a given sequence α. The transformation T^- maps α to the sequence β that contains every second letter of α (so $\beta(t) = \alpha(2t)$). Clearly T^- is not causal. $T^+(\alpha)$ is defined to be the sequence β which repeats each α-letter once; so we have $\alpha(t) = \beta(2t) = \beta(2t+1)$ for all t. This transformation is causal but not computable by a Mealy automaton; an unbounded memory is needed to store for outputs from time $2t$ onwards the relevant α-segment $\alpha(t)\dots\alpha(2t)$, for increasing t. Note that we exclude the possibility to produce outputs of length greater than 1 in one step.

Before we enter the solution of Church's Problem in the framework of automata over infinite sequences, it should be mentioned that an alternative approach has been developed by Rabin [21] via automata on infinite trees. Tree automata allow to deal directly with the space of all sequence pairs (α, β) of input- and output-sequences. In the present paper we pursue the "linear" approach as in [4].

3 From Logic to Games

It is useful to study Church's Problem in the framework of infinite games, following an idea that was proposed by McNaughton [12]. A specification φ defines an infinite two-person game between players A and B who contribute the input-, respectively the output-bits in turn. A play of this game is the sequence of pairs $(\alpha(t), \beta(t))$ of bits supplied for $t = 0, 1, \dots$ by A and B in alternation, and the play $(\alpha(0), \beta(0))\ (\alpha(1), \beta(1))\ (\alpha(2), \beta(2))\ \dots$ is won by player B iff φ is satisfied by the pair (α, β). So the formula φ serves as a *winning condition* (for player B). A Mealy automaton as presented above defines a *winning strategy* for player B in this game; so we speak of a *finite-state winning strategy*.

In this section the game theoretic form of Church's Problem is developed, in two steps: First, the S1S-specifications are transformed into deterministic automata over infinite words ("ω-automata"), and secondly these automata are converted into arenas of infinite games.

3.1 Determinization and Muller Automata

The first step for solving Church's Problem consists of a transformation of a specification $\varphi(\overline{X}, \overline{Y})$ into a semantically equivalent but "operational" form. The

idea is to introduce a finite number of "states" that are visited while a play evolves and at the same time to radically simplify the logical condition to be satisfied. As it turns out, this condition only takes into account which states are visited infinitely often during an infinite play that is built up by players A and B.

This transformation puts Church's Problem into the framework of automata theory. It is remarkable that we do not have any solution of Church's Problem that avoids this transformation at the start – e.g., by an inductive approach of synthesis that is guided by the structure of the original formula φ.

The appropriate model of automaton into which S1S-formulas are to be transformed was introduced by Muller [17] and is called *Muller automaton*. In the present context, a Muller automaton scans deterministically a play $(\alpha(0), \beta(0))$ $(\alpha(1), \beta(1)) \ldots$ as a sequence from $\Sigma = (\{0,1\}^2)^\omega$; the automaton is called equivalent to φ if precisely the plays are accepted that satisfy φ. Its unique run ϱ on a given play between A and B can be viewed as the working of a referee watching the play. The acceptance condition for the run ϱ refers to the "infinity set of ϱ", which is defined as follows (denoting the set of states by Q):

$$\mathrm{Inf}(\varrho) := \{q \in Q \mid \exists^\omega i \ \varrho(i) = q\}$$

The acceptance component of the Muller automaton is a collection $\mathcal{F}$ of state sets, and a run ϱ is declared to be accepting if $\mathrm{Inf}(\varrho)$ belongs to $\mathcal{F}$. Since a set $\mathrm{Inf}(\varrho)$ clearly constitutes a strongly connected subset of the transition graph of the Muller automaton, it suffices to include only strongly connected subsets in $\mathcal{F}$, which we call "accepting loops".

The transformation from S1S to Muller automata can be established by an induction on the construction of S1S-formulas. While the cases of atomic formulas and Boolean operations are straightforward, the quantifier step (without loss of generality regarding the existential second-order quantifier) is of intriguing difficulty. The projection operation involved in an application of the existential quantifier leads immediately to nondeterministic Muller automata, which then have to be determinized.

The determinization problem requires to condense the different runs of a given nondeterministic automaton $\mathcal{A}$ into a single run of a new (deterministic) automaton such that this run allows to decide the existence of a successful run of $\mathcal{A}$. Over a finite word w, acceptance is decided by inspecting the last states of the different $\mathcal{A}$-runs on w. To compute the states reachable by $\mathcal{A}$ via w, it suffices to record the reachable states for each of the prefixes of w. Since the update of this set from one prefix to the next is possible without a reference to previously visited states, the "subset construction" (introduced by Myhill [19] and by Rabin and Scott [22]) suffices, in which a deterministic automaton is built with states that are sets of states of $\mathcal{A}$.

For the determinization of ω-automata one needs additional memory, since repeated visits to certain states on individual runs of the given nondeterministic

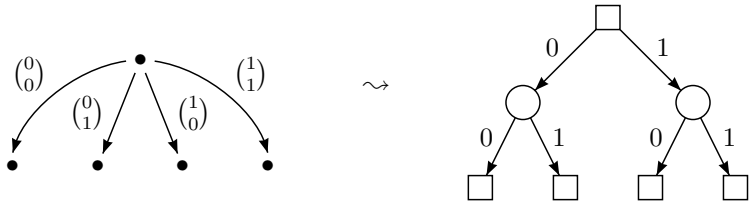
automaton $\mathcal{A}$ have to be recorded. We consider here the case of nondeterministic Büchi automata (to which the case of nondeterministic Muller automata is easily reduced); a run ϱ of a Büchi automaton is called successful if for infinitely many t the state $\varrho(t)$ belongs to a designated set F of accepting states. To check the existence of a successful run of a Büchi automaton deterministically, there are (at least) three types of appropriate memory structures. Such a structure S should be finite, and it should be usable for the test whether an $\mathcal{A}$-run exists on a given ω-word in which a state from F occurs infinitely often; moreover, the test should involve just the information which memory states of S are visited infinitely often and which only finitely often when processing the ω-word under consideration.

The first idea, pursued by McNaughton in [13] and also in the book [28], is to start new computation threads whenever the nondeterminism of $\mathcal{A}$ requires this, to record whether visits of states in F occur, and to devise a policy of merging runs when they reach the same $\mathcal{A}$-state. A different approach is the construction of Muller and Schupp in [18]. Here a version of the run tree of a nondeterministic Büchi automaton is built up while an input word is scanned. The finite prefixes of the run tree (corresponding to the prefixes of the input word) are compressed so that only finitely many different compressed trees can arise. The subset construction is applied in each step, however dividing the reached states into two parts (consisting of the F-states and non-F-states, respectively), which leads to two son nodes of a leaf of the previous tree. A bound on the height and the width of the trees is realized by a compression of paths without branching and by the deletion of double occurrences of a state. A vertex coloring with three colors serves to keep track of repeated visits to states in F. Finally, the celebrated construction of Safra [23] involves a somewhat sparser use of the subset construction; here a new branch of the run tree is opened only when F-states are encountered (so a new son vertex is only created with F-states). Again, a compression policy ensures that the size of the "Safra trees" stays bounded, and a subtle mechanism serves to record repeated visits to F.

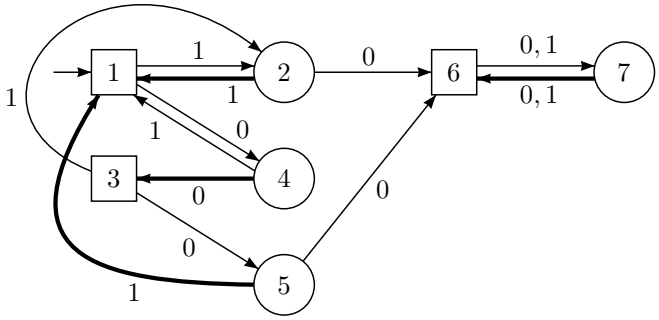
There is no space here to discuss these intriguing constructions in further detail. Even today the subject is not closed; and a major open problem is to devise procedures that substantially reduce (or even minimize) the size of deterministic ω-automata.

3.2 Muller Games

For an analysis of Church's Problem in a game theoretic setting it is useful to distinguish the contribution of bits (in the general case: bit vectors) by the two players A and B. Rather than processing a bit pair $(\alpha(t), \beta(t))$ in one step of the Muller automaton, we introduce two steps, each processing a single bit, and using an intermediate state. Then we have two kinds of states, called A- and B-states. In an A-state, the next bit is to be picked by player A, in a B-state by player B. In a graph theoretical presentation we indicate A-states by boxes and B-states by circles. Thus the transitions of a Muller automaton from a given state are dissolved as follows:



The result is a "game graph". For our example specification above, we obtain the following game graph from a corresponding Muller automaton (the reader should ignore for the moment the boldface notation of some arrows).



The three conditions of our example formula (Sect. 2) can indeed be captured by this graph, by providing an appropriate list of accepting loops. The first condition requires that a bit 1 chosen by A has to be answered by the bit 1 chosen by B. If this is violated (starting from the initial state 1), state 6 (and hence the loop consisting of states 6 and 7) is entered. The second condition says that player B should not pick two zeroes in succession. If this is violated, we would reach 6 and 7 again. We thus exclude states 6 and 7 from the accepting loops. The third condition on fairness means that if A chooses 0 infinitely often (which happens by going to 4 or 5), then B has to choose 0 infinitely often (which is only possible by going from 4 to 3). Altogether we declare a loop F as accepting if it does not contain 6 or 7 and satisfies $(4 \in F \vee 5 \in F \rightarrow 3 \in F)$.

How should player B pick his bits to ensure that the play visits precisely the states of one of these loops F infinitely often? We have to fix how to move from states 2, 4, 5, 7. From 7 player B has to move to 6 since there is no other choice. The other choices can be fixed as follows: From 2 to 1, from 4 to 3, and from 5 to 1 (see boldface arrows). Then, depending on what Player A does, a play starting in 1 will visit infinitely often the states 1 and 2, or the states 1 to 4, or the states 1, 3, 4, 5, or the states 1 to 5. Each of these loops is accepting.

We see that player B has a winning strategy by fixing his moves as stated above. This winning strategy can be converted into a Mealy automaton when we combine again each pair of two successive moves (by player A and then B) into a single transition. We get an automaton with the states 1 and 3 and the following transitions: From 1 via $\binom{1}{1}$ back to 1, from 1 via $\binom{0}{0}$ to 3, and from 3 via $\binom{0}{1}$ and via $\binom{1}{1}$ back to 1. Up to names of states (and the irrelevant initial state) this is precisely the Mealy automaton mentioned in Sect. 2.

In the remainder of the paper, we shall outline a "solution" of Muller games (and some weaker variants like "'weak Muller games"). By a solution we mean two algorithms: The first decides for each state q (A-state or B-state) whether for plays starting in q player B has a winning strategy, and – in this case – the second algorithm allows to construct a Mealy automaton that executes such a winning strategy. In this analysis we may cancel the labels on the transitions. This is motivated by the fact that the winning condition is formulated in terms of visits of states only, regardless of the labels that are seen while traversing edges. When a winning strategy over the unlabelled game graph is to be constructed, it will be easy to re-introduce the labels and use them for a Mealy automaton as required in the original formulation of Church's Problem.

As a preparation, we now summarize the relevant definitions in some more detail.

3.3 Finite-State Games: The Framework

A *game graph* has the form $G = (Q, Q_A, E)$ where $Q_A \subseteq Q$ and $E \subseteq Q \times Q$ is the transition relation. We assume that $\forall q \in Q : qE \neq \emptyset$ (i.e. $\forall q \exists q' : (q, q') \in E$); so plays cannot end in a deadlock (and hence a subset Q_0 of Q induces again a game graph if from each $q \in Q_0$ there is an edge back to Q_0). We set $Q_B := Q \setminus Q_A$. In this paper edges will always lead from Q_A-states to Q_B-states or conversely. A *play* over G from q is an infinite sequence $\varrho = q_0 q_1 q_2 \ldots$ with $q_0 = q$ and $(q_i, q_{i+1}) \in E$ for $i \geq 0$. We assume that player A chooses the next state from a state in Q_A, and player B from a state in Q_B. The set Q will always be finite in the sequel; so we speak of *finite-state games*.

For the formulation of winning conditions, we add a further item to the game graph, depending on the format of the condition. We use either a collection $\mathcal{F} \subseteq 2^Q$ of sets $R \subseteq Q$, or a coloring $c : Q \to \{0, \ldots, k\}$ for some natural number k. In the special case $c : Q \to \{0, 1\}$ we also consider the subset $F = \{q \in Q \mid c(q) = 1\}$ instead. For a collection $\mathcal{F} \subseteq 2^Q$ we introduce two winning conditions. The first is the Muller winning condition mentioned above; it refers to the set of states visited infinitely often in a play ϱ:

$$\mathrm{Inf}(\varrho) := \{q \in Q \mid \exists^\omega i \ \varrho(i) = q\}$$

Player B wins the play ϱ if $\mathrm{Inf}(\varrho) \in \mathcal{F}$. With these conventions we speak of a *Muller game* $(G, \mathcal{F})$. Another use of a system $\mathcal{F}$ leads to the *weak Muller condition* (also called *Staiger-Wagner condition*). Here we refer to the visited states in a play ("occurrence set"):

$$\mathrm{Occ}(\varrho) := \{q \in Q \mid \exists i \ \varrho(i) = q\}$$

Player B wins a play ϱ according to the weak Muller condition if $\mathrm{Occ}(\varrho) \in \mathcal{F}$. We speak of the *weak Muller game* $(G, \mathcal{F})$. From the example in Sect. 2 we obtain a weak Muller game if we delete the third requirement. The items 1 and 2 are captured over the presented game graph (with the states $1, \ldots, 7$) by the condition that none of the states 6 or 7 is ever visited; this is expressed by the weak Muller condition with the system $\mathcal{F}$ that contains all subsets of $\{1, \ldots, 5\}$.

An important special case of weak Muller games is the *reachability game*, given a set $F \subseteq Q$ of states of the game graph (Q, Q_A, E). The reachability condition for player B is satisfied for a play ϱ if some state of ϱ belongs to F. We speak of the *reachability game* (G, F). One obtains an equivalent weak Muller condition by setting $\mathcal{F} = \{R \subseteq Q \mid R \cap F \neq \emptyset\}$. The reachability game (for player B) yields a game with complemented winning condition for player A, namely to stay in the set $Q \setminus F$ throughout. Such a condition is called a *safety condition*. Taking up our example again, we see that the weak Muller condition mentioned above (covering items 1 and 2 of the requirement) amounts to the safety condition, now for player B, to stay in the set $\{1, \ldots, 5\}$ during the whole play.

We now turn to the solution of games, starting with the central concept of strategy. A *strategy for player B from q* is a function $f : Q^+ \to Q$, specifying for any play prefix $q_0 \ldots q_k$ with $q_0 = q$ and $q_k \in Q_B$ some vertex $r \in Q$ with $(q_k, r) \in E$ (otherwise the value of f is chosen arbitrarily). A play $\varrho = q_0 q_1 \ldots$ from $q_0 = q$ is *played according to strategy f* if for each $q_i \in Q_B$ we have $q_{i+1} = f(q_0 \ldots q_i)$. A strategy f for player B from q is called *winning strategy for player B from q* if any play from q which is played according to f is won by player B. In the analogous way, one introduces strategies and winning strategies for player A. We say that A (resp. B) *wins from q* if there is a winning strategy for A (resp. B) from q.

For a game over the graph $G = (Q, Q_A, E)$, the *winning regions of players A and B* are the sets $W_A := \{q \in Q \mid \text{A wins from } q\}$ and $W_B := \{q \in Q \mid \text{B wins from } q\}$. It is obvious that a state cannot belong to both W_A and W_B; so the winning regions W_A, W_B are disjoint. But whether these sets exhaust the whole game graph is a more delicate question. One calls a game *determined* if $W_A \cup W_B = Q$, i.e. from each vertex one of the two players has a winning strategy. Determinacy of infinite games is a central topic in descriptive set theory; with the axiom of choice one can construct games that are not determined. For the games considered in this paper (i.e. games defined in terms of the operators Occ and Inf), determinacy is well-known. Nevertheless we state this claim in the results below, since determinacy is the natural way to show that envisaged winning strategies are complete: In order to show that the domain D of a strategy covers the entire winning region of one player, one verifies that from each state outside D the other player has a winning strategy.

To "solve" a game over the graph $G = (Q, Q_A, E)$ involves two tasks:

1. to decide for each $q \in Q$ whether $q \in W_B$ or $q \in W_A$,
2. and depending on q to construct a suitable winning strategy from q (for player B, respectively A).

For item 2 two kinds of strategies will be employed, the memoryless and the finite-state strategies. A strategy $f : Q^+ \to Q$ is *memoryless* if the value of $f(q_1 \ldots q_k)$ only depends on the "current state" q_k. For the definition of finite-state strategies, we first observe that over a finite set Q, a strategy $f : Q^+ \to Q$ can be considered as a word function. We say that f is a *finite-state strategy* if it is computed by a Mealy automaton. In the present context we use the format $\mathcal{S} = (S, Q, Q, s_0, \delta, \tau)$ with state set S, input alphabet Q, output alphabet Q, initial

state s_0, transition function $\delta : S \times Q \to S$, and output function $\tau : S \times Q_A \to Q$ for player A (respectively $\tau : S \times Q_B \to Q$ for player B). The *strategy $f_{\mathcal{S}}$ computed by $\mathcal{S}$* is now defined by $f_{\mathcal{S}}(q_0 \dots q_k) = \tau(\delta^*(s_0, q_0 \dots q_{k-1}), q_k)$ (where $\delta^*(q, w)$ is the state reached from q after processing the input word w and τ is chosen for the player under consideration).

Now we state the main theorem on weak Muller games and Muller games.

Theorem 1. *Weak Muller games and Muller games are determined, and for a weak Muller game, respectively Muller game $(G, \mathcal{F})$ one can effectively compute the winning regions of the two players, and one can construct, for each state q of G, a finite-state winning strategy from q for the respective winning player.*

The part concerning Muller games is the Büchi-Landweber Theorem and gives the desired solution of Church's Problem. For this, one proceeds as in the previous section, i.e. one transforms a given S1S-formula φ into a Muller automaton $\mathcal{M}$ which is then converted to a game graph G with Muller winning condition. Note that the game graph G inherits an initial state from $\mathcal{M}$. Using the Büchi-Landweber Theorem, one checks whether this initial state belongs to the winning region of player B, and in this case one obtains a Mealy automaton $\mathcal{S}$ that realizes a winning strategy from the initial state. The desired finite-state strategy for the original formula φ is then easily constructed as a product automaton from $\mathcal{M}$ and $\mathcal{S}$. Its memory thus combines the state space of the Muller automaton $\mathcal{M}$ with that of the strategy automaton $\mathcal{S}$. It is not yet well understood how these two aspects play together in general. Our example in Sect. 2 illustrates the case that in addition to the states of $\mathcal{M}$ no additional memory is necessary.

3.4 Reachability Games

As a preparatory step for Theorem 1 we solve reachability games. Recall that a reachability game (G, F) involves the winning condition (for player B) that the play should reach somewhere a state of the set F. The solution relies on a simple backward search of the game graph, starting with the set F.

Theorem 2. *A reachability game (G, F) with $G = (Q, Q_A, E)$ and $F \subseteq Q$ is determined, and the winning regions W_A, W_B of players A and B, respectively, are computable, as well as corresponding memoryless winning strategies.*

Proof. We compute, for $i = 0, 1, \dots$, the vertices from which player B can force a visit in F within i moves. We call this set the i-th "attractor" (for B):

$$\begin{aligned}\mathrm{Attr}_B^i(F) := \{&q \in Q \mid \text{from } q \text{ player B can force a visit of } F \\ &\text{in} \le i \text{ moves}\}\end{aligned}$$

Its computation for increasing i is known from the theory of finite games (and corresponds to the well-known analysis of AND-OR-trees):

$$\begin{aligned}\mathrm{Attr}_B^0(F) &= F, \\ \mathrm{Attr}_B^{i+1}(F) &= \mathrm{Attr}_B^i(F) \\ &\quad \cup \{q \in Q_B \mid \exists (q, r) \in E : r \in \mathrm{Attr}_B^i(F)\} \\ &\quad \cup \{q \in Q_A \mid \forall (q, r) \in E : r \in \mathrm{Attr}_B^i(F)\}\end{aligned}$$

So for step $i+1$ we include a state of Q_B if from it some edge can be chosen into $\mathrm{Attr}_B^i(F)$. We can fix such a choice for each Q_B-state in $\mathrm{Attr}_B^{i+1}(F)$ ($i = 0, 1, \ldots$) in order to build up a memoryless strategy. We include a state in Q_A in $\mathrm{Attr}_B^{i+1}(F)$ if all edges from it lead to $\mathrm{Attr}_B^i(F)$. The sequence $\mathrm{Attr}_B^0(F) \subseteq \mathrm{Attr}_B^1(F) \subseteq \mathrm{Attr}_B^2(F) \subseteq \ldots$ becomes stationary for some index k since Q is finite. Since $k \leq |Q|$ we can define $\mathrm{Attr}_B(F) := \bigcup_{i=0}^{|Q|} \mathrm{Attr}_B^i(F)$.

Later we shall also use the set $\mathrm{Attr}_A(F)$, defined in the analogous way for player A.

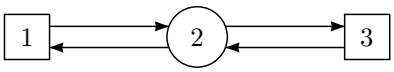
With the inductive construction it was explained that $\mathrm{Attr}_B(F) \subseteq W_B$; furthermore we have defined a uniform memoryless winning strategy which can be applied to any state in W_B regardless of the start of the play. (For states in $Q_B \cap F$ the choice of the next state is arbitrary.)

For the converse inclusion $W_B \subseteq \mathrm{Attr}_B(F)$ we have to show that $\mathrm{Attr}_B(F)$ exhausts the winning region W_B. For this, we show that from each state in the complement of $\mathrm{Attr}_B(F)$, player A has a winning strategy (which is again memoryless). It suffices to verify that from any state q in $Q \setminus \mathrm{Attr}_B(F)$ player A can force to stay outside $\mathrm{Attr}_B(F)$ also in the next step. This is checked by a case distinction: If $q \in Q_A$, there must be an edge back into $Q \setminus \mathrm{Attr}_B(F)$, otherwise all edges from q would go to $\mathrm{Attr}_B(F)$ whence q would belong to $\mathrm{Attr}_B(F)$. If $q \in Q_B$, all edges from q must lead to $Q \setminus \mathrm{Attr}_B(F)$, because otherwise there would be an edge to $\mathrm{Attr}_B(F)$ and q would again belong to $\mathrm{Attr}_B(F)$.

4 Appearance Records and Game Simulations

4.1 Appearance Records

For Muller games, both in the weak and the unrestricted form, memoryless strategies are not enough. A simple example illustrates this. Consider the following game graph G and the set $\mathcal{F} = \{\{1, 2, 3\}\}$.



The weak Muller game $(G, \mathcal{F})$ requires for player B to visit all states in order to win. From vertex 2 there is no memoryless winning strategy: Neither the choice to move to 1 nor the choice to move to 3 will ensure to reach each vertex. On the other hand, a one-bit memory will do: When coming back to 2 we should know whether 1 or 3 was visited before, and then we should move to 3, respectively 1 (and maybe do this perpetually from that moment onwards). A general principle derivable from this solution is to "remember where we have been already". This principle corresponds to a simple experience of every-day life: When there is a task ahead consisting of several items, keep a list of what was done already (and thus of what still has to be done).

For the strong Muller game $(G, \mathcal{F})$, both vertices 1 and 3 have to be visited again and again. Clearly it does not suffice just to remember where we have been already: After the visits of 1 and 3 it is necessary to switch from 1 to 3 and back

again and again. The natural solution is to "remember where we went last time" – and then to do the choice accordingly, going to the respective "other" vertex.

In the first case (of weak Muller games), we are led to a memory structure that allows to store in an accumulative way the vertices that were already visited in a play. Given a weak Muller game $(G, \mathcal{F})$ with $G = (Q, Q_A, E)$ and $\mathcal{F} \subseteq 2^Q$, we define the transition structure of an automaton $\mathcal{S}$ with the power set 2^Q of Q as its set of states and Q as its input alphabet. Having read the input word $q_1 \dots q_k$, its state will be $\{q_1, \dots, q_k\}$. So the initial state is $\emptyset$ and the transition function $\delta : 2^Q \times Q \to 2^Q$ is defined by $\delta(R, p) = R \cup \{p\}$. This memory of subsets of Q with the mentioned update rule is called *appearance record*. We shall show that this memory structure suffices for winning strategies in arbitrary weak Muller games over G. What remains is to fix the output function for $\mathcal{S}$.

Let us now treat the case of (strong) Muller games. Our example above motivates to keep a refined record of the states visited in a play, taking into account the order in which states were "visited last time". A naive way to realize this kind of memory is to arrange the set of visited states in a list where the first entry is the currently visited state q, the second one the state $q' \neq q$ visited last before q, the third one the state q'' different from q, q' visited last before q, q', and so on until the set of previously visited states is exhausted. It will be useful to work with a slight (but essential) refinement of this list structure, which goes back to McNaughton [12] and is today known as *latest appearance record*, short "LAR".

Consider a Muller game $(G, \mathcal{F})$ with $G = (Q, Q_A, E)$ and $Q = \{1, \dots, n\}$. A LAR is a pair $((i_1, \dots, i_r), h)$ where the i_j are distinct states from Q and $0 \leq h \leq r$. Following Büchi, we call the index h the *hit* of the LAR. Again we define the transition structure of an automaton $\mathcal{S}$, now with the set of LAR's over Q as its set of states. The initial state is $((\,), 0)$ (empty list and hit 0). The transition function changes a given LAR upon input $q \in Q$ by listing the new state q at the front: If it was not present in the previous LAR, then it is added (and h is set to be 0); if it occurs in the previous LAR, then it is shifted to the front and the position where it was taken from is the value of h.

We give an example for a set Q of four states, which we name A, B, C, D to avoid confusion with the hit values $1, \dots, 4$. Also we indicate the hit value h by underlining the h-th position of the corresponding list (if $h > 0$). Suppose a play ϱ starts with the states $A, C, C, D, B, D, C, D, D, \dots$. Then we obtain the following sequence of LAR's (where we skip the initial LAR $((\,), 0)$):

Suppose that the play goes on only by states C and D (and both are chosen again and again). Then the states A, B will not be touched anymore, and the hit will assume 2 as maximal value thereafter again and again: It will no more be 3 or 4 (since A, B stay where they are), and it cannot finally stay with value 1 (since then only a single state, namely the leading one of the LAR, would be visited from some point onwards). The maximal hit visited infinitely often thus indicates the cardinality of the set $\mathrm{Inf}(\varrho)$.

Visited state	Reached LAR
A	(A)
C	(CA)
C	$(\underline{C}A)$
D	(DCA)
B	$(BDCA)$
D	$(D\underline{B}CA)$
C	$(CD\underline{B}A)$
D	$(D\underline{C}BA)$
D	$(\underline{D}CBA)$

Let us summarize the definition of the automaton $\mathcal{S} = (S, Q, s_0, \delta)$ which realizes, given a play prefix $i_1 \dots i_k \in Q^*$, the computation of the resulting LAR. The state set S is the set of LAR's over Q, we have $s_0 = (( ), 0)$, and the transition function $\delta : S \times Q \to S$ realizes the update of the LAR as follows: We have $\delta(((i_1 \dots i_r), h), i) = ((ii_1 \dots i_r), 0)$ if i does not occur in $(i_1 \dots i_r)$; otherwise, if $i = i_k$ cancel i from $(i_1 \dots i_r)$ to obtain $(j_1 \dots j_{r-1})$ and set $\delta(((i_1 \dots i_r), h), i) = ((ij_1 \dots j_{r-1}), k)$.

Note that we obtain the (simple) appearance record if we discard the order in the state-lists and delete the h-value. We shall show that the LAR memory structure over Q will suffice for realizing winning strategies in Muller games over Q. Again, it only remains to supply the output function τ for the automaton $\mathcal{S}$ in order to obtain the complete definition of a Mealy automaton.

We add a historical remark. The paper [12] in which McNaughton introduced the fundamental data structure of LAR is a technical report that was not published as a journal paper (it contained an error in an attempted solution of Muller games). McNaughton used the name "order-vector"; the term "latest appearance record" (LAR) was introduced by Gurevich and Harrington in their influential work [10] on automata over infinite trees.

4.2 Game Simulations and Parity Conditions

The two versions of appearance record introduced in the previous section allow to reformulate the winning conditions (weak Muller condition, strong Muller condition) in a form that makes the solutions of the corresponding games much easier.

First let us consider weak Muller games. For a play ϱ, consider the sequence of associated appearance records, as assumed in the run of the Mealy automaton that realizes the necessary updates. The set of visited states increases weakly monotonically during the play and finally reaches the value $\mathrm{Occ}(\varrho)$ on which it stays fixed. Similarly the cardinality of the set of visited states increases until it reaches the value $|\mathrm{Occ}(\varrho)|$. This observation enables us to express the weak Muller winning conditon "$\mathrm{Occ}(\varrho) \in \mathcal{F}$" in different way. We associate a number $c(R)$ with each subset R of Q, also called its color, which conveys two informations: the size of R, and whether R belongs to $\mathcal{F}$ or not. In the first case, we take the even color $2 \cdot |R|$, otherwise the odd color $2 \cdot |R| - 1$ (assuming $R \neq \emptyset$):

$$c(R) := \begin{cases} 2 \cdot |R| & \text{if } R \in \mathcal{F} \\ 2 \cdot |R| - 1 & \text{for } R \notin \mathcal{F} \end{cases}$$

For $R = \emptyset$ let $c(R) = 0$. – The following claim is then obvious:

Remark 3. Let ϱ be a play and $R_0, R_1, R_2, \ldots$ be the sequence of the associated appearance records. Then $\mathrm{Occ}(\varrho) \in \mathcal{F}$ iff the maximal color in the sequence $c(R_0)c(R_1)c(R_2)\ldots$ is even.

This remark motivates a new winning condition over game graphs $G = (Q, Q_A, E)$ that are equipped with a coloring $c : Q \to \{0, \ldots, k\}$. The *weak parity condition* with respect to coloring c says: Player B wins the play $\varrho = r_0 r_1 r_2 \ldots$ iff the maximum color in the sequence $c(r_0)c(r_1)c(r_2)\ldots$ is even. Given a game graph G and a coloring c with the weak parity winning condition, we speak of the *weak parity game* (G, c).

Using this, one transforms a weak Muller game $(G, \mathcal{F})$ into a weak parity game (G', c): Given $G = (Q, Q_A, E)$ let $G' = (2^Q \times Q, 2^Q \times Q_A, E')$ where $((P, p), (R, r)) \in E'$ iff $(p, r) \in E$ and $R = P \cup \{p\}$, and let $c(R, r) := 2 \cdot |R|$ if $R \in \mathcal{F}$, otherwise $2 \cdot |R| - 1$. Each play $\varrho = r_0 r_1 \ldots$ in G induces the play $\varrho' = (\emptyset, r_0)(\{r_0\}, r_1)\ldots$ in G', which is built up according to the definition of E'. We have by construction that ϱ satisfies the weak Muller condition w.r.t. $\mathcal{F}$ iff ϱ' satisfies the weak parity condition w.r.t. c.

This transformation of $(G, \mathcal{F})$ into (G', c) (with a change of the winning condition) is a "game simulation". (We skip a general definition since we only apply it for the present case and the case of Muller games.)

The simulation has an interesting consequence when the latter game (the weak parity game) allows memoryless winning strategies. Namely, a memoryless strategy over G' immediately determines the output function for the Mealy automaton that computes the appearance records: If the memoryless strategy (say for player B) requires to proceed from position (R, q) (where $q \in Q_B$) to (R', q'), then the output function value $\tau(R, q)$ of the Mealy automaton is set to be q' (and the new state is $R' = R \cup \{q\}$). Also the decision whether a state q of G belongs to the winning region of B is provided by the analysis of the corresponding weak parity game over G', since for each state of a weak parity game we shall determine the winner. Applying this to the state $(\emptyset, q)$ of G' we obtain the answer also for q in the graph G.

In the next section we shall show that weak parity games can indeed be solved with memoryless winning strategies. Using the previous remark this completes the solution of weak Muller games in Theorem 1.

Let us turn to the case of Muller games. We proceed as before, now using the latest appearance record structure LAR in place of the appearance record. Consider a play ϱ over Q and the associated sequence ϱ' of LAR's. We collect the entries of a LAR $((i_1 \ldots i_r), h)$ up to position h as the *hit set* $\{i_1, \ldots, i_h\}$ of the LAR. If h is the maximal hit assumed infinitely often in ϱ', we may pick a position (time instance) in ϱ' where no unlisted state enters any more later in the play and where only hit values $\leq h$ occur afterwards. From that point

onwards the states listed after position h stay fixed, and thus also the hit set for the hit value h stays fixed. We call this set *the hit set for the maximal hit occurring infinitely often in* ϱ'. The following statement is now easily verified:

Remark 4. Let ϱ be a sequence over Q and ϱ' be the associated sequence of LAR's. The set $\mathrm{Inf}(\varrho)$ coincides with the hit set H for the maximal hit h occurring infinitely often in ϱ'.

For the proof, consider the point in ϱ from where no new states will occur and where all visits of states that are visited only finitely often are completed. After a further visit of all the states in $\mathrm{Inf}(\varrho)$, these states will stay at the head of the LAR's (in various orders), and the hit values will be $\leq k := |\mathrm{Inf}(\varrho)|$. It remains to show that the hit value in ϱ' reaches k again and again (so that k is the maximal hit occurring infinitely often in ϱ'). If the hit was $< k$ from some point onwards, the state q listed on position k would not be visited later and thus not be in $\mathrm{Inf}(\varrho)$.

Remark 4 allows to transform the Muller winning condition for a play ϱ into a different winning condition applied to the associated play ϱ'. By Remark 4 we know that the Muller winning condition holds for the play ϱ iff *the hit set for the maximal hit occurring infinitely often in* ϱ' *belongs to* $\mathcal{F}$. This allows us to extract two data from the LAR's which are sufficient to decide whether the play ϱ satisfies the Muller condition: the hit value and the information whether the corresponding hit set belongs to $\mathcal{F}$. We combine these two data in the definition of a coloring of the LAR's. Define

$$c(((i_1 \dots i_r), h)) := \begin{cases} 2h & \text{if } \{i_1, \dots, i_h\} \in \mathcal{F} \\ 2h-1 & \text{if } \{i_1, \dots, i_h\} \notin \mathcal{F} \end{cases}$$

for $h > 0$ and let $c(((i_1 \dots i_r), 0)) = 0$. Then the Muller condition $\mathrm{Inf}(\varrho) \in \mathcal{F}$ is satisfied iff the maximal color occurring infinitely often in $c(\varrho'(0))c(\varrho'(1)) \dots$ is even. This is a "parity condition" (as introduced by Mostowski [16] and Emerson and Jutla [9][2]). The only difference to the weak parity condition is the reference to colors occurring infinitely often rather than to those which occur at all.

In general, the parity condition refers to a coloring $c : Q \to \{0, \dots, k\}$ of a game graph G; it is the following requirement on a play ϱ:

$$\bigvee_{j \text{ even}} (\exists^\omega i : c(\varrho(i)) = j \quad \wedge \neg \exists^\omega i : c(\varrho(i)) > j)$$

The pair (G, c) with this convention for the winning condition for player B is called a *parity game*.

In complete analogy to the case of weak Muller games, one can set up a game simulation of a Muller game $(G, \mathcal{F})$ by a parity game (G', c). A state of G' is a pair consisting of a LAR ℓ and a state q from Q. An edge is introduced from (ℓ, q)

[2] Other names appearing in the literature are "Mostowski condition" and "Rabin chain condition".

to (ℓ', q') if the edge (q, q') exists in G and ℓ' results from ℓ by the LAR-update that lists q at the head. A play ϱ over G then corresponds to a play ϱ' over G'. The coloring c is defined as above.

We shall show that parity games can be solved with memoryless winning strategies. As for the case of weak Muller and weak parity games, a memoryless winning strategy of player B in the parity game over G' yields a finite-state winning strategy of player B in the Muller game over G. The decision whether for a state q of G such a winning strategy exists is done by testing whether player B wins from position $((( ), 0), q)$ in the parity game over G'.

5 Solving Weak and Strong Parity Games

A central difficulty in the solution of weak Muller games and Muller games is the possibly complicated structure of the system $\mathcal{F}$ of "winning state-sets". The first proof of the Büchi-Landweber Theorem [4] involves an intriguing analysis of the partial order (by set inclusion) of the power set of the set Q of states of the game graph. The transformation to a game with a (weak or strong) parity condition stratifies the winning condition by introducing the total order of colors. As we shall see, this order can be exploited for an inductive construction, again for both the weak parity games and the parity games. In both cases, an iterated application of attractor computations suffices; thus, the game solution ultimately rests on simple reachability tests.

Theorem 5. *A weak parity game (G, c) is determined, and one can compute the winning regions W_A, W_B and also construct corresponding memoryless winning strategies for the players A and B.*

It may be noted that we suppressed the initial states q when speaking about memoryless winning strategies. In the proof we shall see that – as for reachability games – the strategies can be defined independently of the start state (as long as it belongs to the winning region of the respective player).

Proof. Let $G = (Q, Q_A, E)$ be a game graph (we do not refer to the special graph G' above), $c : Q \to \{0, \ldots, k\}$ a coloring (w.l.o.g. k even, otherwise switch players). Set $C_i = \{q \in Q \mid c(q) = i\}$.

We first compute the attractor for B of the states with maximal color, which is even. When player B reaches such a state the play is won whatever happens later. So $A_k := \mathrm{Attr}_B(C_k)$ is a part of the winning region of player B.

The remaining vertices form the set $Q \setminus A_k$; the subgraph induced by $Q \setminus A_k$ in G is again a game graph. (Note that from each state q in $Q \setminus A_k$ there is at least one edge back to $Q \setminus A_k$, otherwise – as seen by case distinction whether $q \in Q_A$ or $q \in Q_B$ – q would belong to $A_k = \mathrm{Attr}_B(C_k)$.)

Now in the subgame over $Q \setminus A_k$ we compute $A_{k-1} := \mathrm{Attr}_A(C_{k-1} \setminus A_k)$; from these vertices player A can reach the highest odd color $k-1$ and guarantee to stay away from A_k, in the same way as explained above for reachability games (see Sect. 4.1).

In both sets we can choose memoryless winning strategies, over A_k for B, and over A_{k-1} for A. In this way we continue to adjoin "slices" of the game graph, taking B- and A-attractors in alternation, in order to obtain the winning regions of B and A. The next set A_{k-2} is the set of all states $q \in Q \setminus (A_{k-1} \cup A_k)$ from which player B can force the play to $C_{k-2} \setminus (A_{k-1} \cup A_k)$. We denote this set by $\mathrm{Attr}_B^{Q \setminus (A_{k-1} \cup A_k)}(C_{k-2} \setminus (A_{k-1} \cup A_k))$. The exponent indicates the (domain of) the game graph in which the attractor computation takes place. In order to facilitate the notation for the general case, set $Q_i := Q \setminus (A_{i+1} \cup \ldots \cup A_k)$.

So we compute the sets $A_k, A_{k-1}, \ldots, A_0$ inductively as follows:

$$\begin{aligned} A_k &:= \mathrm{Attr}_B(C_k) \\ A_{k-1} &:= \mathrm{Attr}_A^{Q_{k-1}}(C_{k-1} \setminus A_k) \end{aligned}$$

and for $i = k-2, \ldots, 0$:

$$A_i := \begin{cases} \mathrm{Attr}_B^{Q_i}(C_i \setminus (A_{i+1} \cup \ldots \cup A_k)) & \text{if } i \text{ even} \\ \mathrm{Attr}_A^{Q_i}(C_i \setminus (A_{i+1} \cup \ldots \cup A_k)) & \text{if } i \text{ odd} \end{cases}$$

The memoryless strategies for A and B are chosen as explained for the initial cases A_k, A_{k-1}. Now we have

$$W_B = \bigcup_{i \text{ even}} A_i \quad \text{and} \quad W_A = \bigcup_{i \text{ odd}} A_i$$

For the correctness, one verifies by induction on $j = 0, \ldots, k$:

$$\bigcup_{\substack{i=k-j \\ i \text{ even}}}^{k} A_i \subseteq W_B \qquad \bigcup_{\substack{i=k-j \\ i \text{ odd}}}^{k} A_i \subseteq W_A$$

Returning to the solution of weak Muller games, we note that a finite-state winning strategy can be realized with 2^n memory states over a game graph with n states, due to the introduction of appearance records.

Let us turn to the case of parity games, following a proof of McNaughton [14].

Theorem 6. *A parity game (G, c) is determined, and one can compute the winning regions W_A, W_B and also construct corresponding memoryless winning strategies for the players A and B.*

Proof. Given $G = (Q, Q_A, E)$ with coloring $c : Q \to \{0, \ldots, k\}$ we proceed by induction on $|Q|$, the number of states of G.

The induction start (Q is a singleton) is trivial. In the induction step assume that the maximal color k is even (otherwise switch the roles of players A and B). Let q be a state of the highest (even) color k and define $A_0 = \mathrm{Attr}_B(\{q\})$. As the complement of an attractor, the set $Q \setminus A_0$ defines a subgame. The induction hypothesis applied to the game over the subgraph induced by $Q \setminus A_0$ ensures a partition of $Q \setminus A_0$ into the winning regions U_A, U_B of the two players (with corresponding memoryless winning strategies) in the game over $Q \setminus A_0$.

We now distinguish two cases:

1. From q, player B can ensure to be in $U_B \cup A_0$ in the next step,
2. From q, player A can ensure to be in U_A in the next step.

Let us first verify that one of the two cases applies (which gives a kind of local determinacy). Assume Case 1 fails. If $q \in Q_B$, then all transitions from q have to go to U_A, otherwise we would be in Case 1. By the same reason, if $q \in Q_A$, then some transition from q goes to U_A; so Case 2 applies.

In Case 1, one shows that the winning region W_B of B in G is $U_B \cup \mathrm{Attr}_B(\{q\})$ and that $W_A = U_A$. For player B, the memoryless winning strategy is composed of the memoryless strategy over U_B by induction hypothesis in the game over $Q \setminus A_0$, of the attractor strategy over $\mathrm{Attr}_B(\{q\})$, and possibly of the edge choice in q according to "Case 1"; for player A just the memoryless strategy over U_A is taken. For the claim $U_B \cup \mathrm{Attr}_B(\{q\}) \subseteq W_B$ note that a play from a state in $U_B \cup \mathrm{Attr}_B(\{q\})$ either remains in U_B from some point onwards, whence Player B wins by induction hypothesis, or it visits (due to moves of player A) the attractor A_0 and hence q again and again, so that player B wins by seeing the highest color (even!) repeatedly. The claim $U_A \subseteq W_A$ is clear by induction hypothesis.

We turn to Case 2. From our analysis above we know that $q \in \mathrm{Attr}_A(U_A)$. We consider the set $A_1 = \mathrm{Attr}_A(U_A \cup \{q\})$, clearly of cardinality ≥ 1. So we can apply the induction hypothesis to the domain $Q \setminus A_1$. We obtain a partition of this domain into winning regions V_A, V_B for A and B in the subgame over $Q \setminus A_1$, with corresponding memoryless winning strategies. Now it is easy to verify $W_B = V_B$ and $W_A = V_A \cup A_1$; memoryless winning strategies for B, respectively A, are provided by the induction hypothesis and by the attractor strategy over A_1.

Finally we note that the inductive construction can be turned into a recursive procedure which produces, given G and the coloring c, the desired winning regions and memoryless strategies.

The recursive procedure appearing in this proof involves a nested call of the inductive hypothesis, which means that for each induction step the computational effort doubles, resulting in an overall exponential runtime. It is known that the problem "Given a parity game (G, c) and a state q, does q belong to the winning region of B?" is in the complexity class NP $\cap$ co-NP. Whether this problem is decidable in polynomial time is one of the major open problems in the algorithmic theory of infinite games.

For the memory size of Mealy automata that realize winning strategies, we obtain a higher bound than for weak Muller games. Over a graph with n states the bound $n! \cdot n$ states suffices. This bound can be met by simplifying the LAR construction introduced above, in the sense that only state lists of length n (and no shorter lists) are used. It is easy to adapt our construction to this format. That the factorial function also provides a lower bound for the memory size was shown in [7].

6 Conclusion

We have presented an approach to Church's Problem that involves three basic ingredients, namely determinization, the stratification of Muller and weak Muller games by different versions of appearance records, and an iterated application of simple reachability tests in the solution of games.

Today, we see that Church's Problem was the starting point for a highly active area of research in computer science, in the last 20 years even with a great influence in practical verification and program synthesis. Thus the "old" parts of automata theory for infinite computations, as addressed by Boaz Trakhtenbrot in the Preface of the book [28], turned out extremely fruitful. It seems certain that the vision of "new trends" as proposed in [28] already decades ago will lead to many more results that share both beauty and an even wider range of applicability.

Acknowledgment

I thank the editors, in particular Alex Rabinovich, for their encouragement and patience.

References

1. Büchi, J.R.: Weak second-order arithmetic and finite automata. Z. Math. Logik Grundlagen Math. 6, 66–92 (1960)
2. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Nagel, E., et al. (eds.) Proc. 1960 International Congress on Logic, Methodology and Philosophy of Science, pp. 1–11. Stanford University Press (1962)
3. Büchi, J.R., Elgot, C.C.: Decision problems of weak second-order arithmetics and finite automata, Abstract 553-112, Notices Amer. Math. Soc. 5, 834 (1958)
4. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies, Trans. Trans. Amer. Math. Soc 138, 367–378 (1969)
5. Church, A.: Applications of recursive arithmetic to the problem of circuit synthesis. In: Summaries of the Summer Institute of Symbolic Logic, vol. I, pp. 3–50. Cornell Univ, Ithaca, N.Y (1957)
6. Church, A.: Logic, arithmetic, and automata. In: Proc. Int. Congr. Math. 1962, Inst. Mittag-Leffler, Djursholm, Sweden, pp. 23–35 (1963)
7. Dziembowski, S., Jurdziński, M., Walukiewicz, I.: How much memory is needed to win infinite games? In: Proc. 12th IEEE Symp. on Logic in Computer Science, pp. 99–110. IEEE Computer Society Press, Los Alamitos (1997)
8. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Trans. Amer. Math. Soc. 98, 21–52 (1961)
9. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus, and determinacy. In: Proc. 32nd FoCS 1991, pp. 368–377. IEEE Comp. Soc. Press, Los Alamitos (1991)
10. Gurevich, Y., Harrington, L.: Trees, automata, and games. In: Proc. 14th ACM Symp. on the Theory of Computing, pp. 60–65. ACM Press, New York (1982)
11. Hopcroft, J.E., Ullman, J.D.: Formal Languages and Their Relation to Automata. Addison-Wesley, Boston (1969)

12. McNaughton, R.: Finite-state infinite games, Project MAC Rep. MIT, Cambridge (1965)
13. McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Inf. Contr. 9, 521–530 (1966)
14. McNaughton, R.: Infinite games played on finite graphs. Ann. Pure Appl. Logic 65, 149–184 (1993)
15. Moore, E.F. (ed.): Sequential Machines – Selected Papers. Addison-Wesley, Reading, Mass (1963)
16. Mostowski, A.W.: Regular expressions for infinite trees and a standard form of automata. In: Skowron, A. (ed.) SCT 1984. LNCS, vol. 208, pp. 157–168. Springer, Heidelberg (1985)
17. Muller, D.E.: Infinite sequences and finite machines. In: Proc. 4th IEEE Ann. Symp. on Switching Circuit Theory and Logical Design, pp. 3–16. IEEE Press, Los Alamitos (1963)
18. Muller, D.E., Schupp, P.E.: Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the results of Rabin, McNaughton, and Safra. Theor. Comput. Sci. 141, 69–107 (1995)
19. Myhill, J.: Finite automata and the representation of events, WADC Tech. Rep. 57-624, pp. 112-137 (1957)
20. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. 141, 1–35 (1969)
21. Rabin, M.O.: Automata on infinite objects and Church's Problem, Amer. Math. Soc., Providence RI (1972)
22. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Develop. 3, 114–125 (1959)
23. Safra, S.: On the complexity of omega-automata. In: Proc. 29th Ann. Symp.on Foundations of Computer Science, White Plains, New York, pp. 319–327. IEEE Computer Society Press, Los Alamitos (1988)
24. Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (1995)
25. Thomas, W.: Solution of Church's Problem: A tutorial. In: Apt, K., van Rooij, R. (eds.) New Perspectives on Games and Interaction, vol. 5, Amsterdam Univ. Press, Texts on Logic and Games (to appear)
26. Trakhtenbrot, B.A.: On operators realizable in logical nets. Dokl. Akad. Naut. SSSR 112, 1005–1007 (1957) (in Russian)
27. Trakhtenbrot, B.A.: Synthesis of logical nets whose operators are described of monadic predicates. Dokl. Akad. Naut. SSSR 118, 646–649 (1958) (in Russian)
28. Trakhtenbrot, B.A., Barzdin, Ya.M.: Finite Automata. Behavior and Synthesis. North-Holland, Amsterdam (1973)