Solution of Church's Problem: A Tutorial

Wolfgang Thomas

[1] Lehrstuhl Informatik 7
RWTH Aachen University
52074 Aachen, Germany
thomas@informatik.rwth-aachen.de

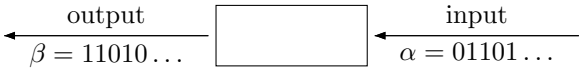
Abstract

Church's Problem (1957) asks for the construction of a finite-state procedure that transforms any input sequence α letter by letter into an output sequence β such that the pair (α, β) satisfies a given specification. Even after the solution by Büchi and Landweber in 1969 (for specifications in monadic second-order logic over the structure $(\mathbb{N}, +1)$), the problem has stimulated research in automata theory for decades, in recent years mainly in the algorithmic study of infinite games. We present a modern solution which proceeds in several stages (each of them of moderate difficulty) and provides additional insight into the structure of the synthesized finite-state transducers.

1 Introduction

Fifty years ago, during the "Summer Institute of Symbolic Logic" at Cornell University in 1957, Alonzo Church considered in [3] a problem which is both simply stated and fundamental.

Imagine a scenario in which an infinite bit stream α is to be transformed, bit by bit, into an infinite stream β, as indicated in the following figure.



The task is to construct a finite-state procedure for this transformation when we are given a "specification" of the relation between α and β. This specification is usually presented as a formula of a logical system. In short words: We have to fill the box, given a description of the desired relation R between input α and output β. The problem is a question on automatic program synthesis which suprisingly can be answered positively when the specification language is not too expressive.

This setting for program synthesis is fundamentally different from the classical framework in which terminating programs for data transformations

are considered. For correctness of a terminating program one relates the data given to the program before the start of the computation to those produced by the program at the end of the computation. Usually the data are from an infinite domain like the natural numbers. In Church's Problem, we deal with non-terminating computations in which inputs and outputs are interleaved, and the aspect of infinity enters in the dimension of time. On the other hand, the data processed in a single step are from a finite domain (in our example just $\{0,1\}$). It is this shift of infinity from data to time that allows to avoid undecidability results as known from the verification (or even synthesis) of terminating programs over infinite data spaces.

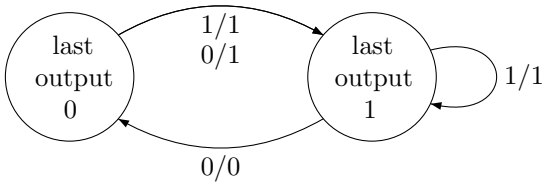
Let us look at an example. The relation R is defined by the conjunction of three conditions on the input-output stream (α, β). We use self-explanatory notation: $\alpha(t)$ is the t-th bit of α $(t = 0, 1, \ldots)$, and $\exists^\omega$ is the quantifier "there exist infinitely many".

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 first two conditions are satisfied easily by producing output 1 at each moment. 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 finite-state transducer displayed below. It is presented as an automaton in which each transition is labelled with an input bit and the corresponding output bit. As initial state we take, for example, the left-hand state.



For a more precise statement of Church's Problem, it is necessary to fix the format of the specifications and that of the "solutions". Let us first address the solutions. Among the many concepts of transformations of sequences, only a very special form is admitted for Church's Problem. Two aspects are relevant, the requirement of a computation "bit by bit", and the restriction to "finite-state" solutions. The first requirement means that the output bit $\beta(t)$ has to be produced without delay after receipt of the input bit $\alpha(t)$. Thus $\beta(t)$ can depend only on the input bits up to time t, i.e. on the input prefix $\alpha(0)\dots\alpha(t)$. This is a sharper restriction than that of "continuity" (in the Cantor topology over $\{0,1\}^\omega$), which would mean that $\beta(t)$ depends on some finite prefix of α – possibly $\alpha(0)\dots\alpha(s)$ with $s > t$. As an illustration, consider the transformation T_1 with $T_1(\alpha) = \alpha(0)\alpha(2)\alpha(4)\dots$. It is continuous but excluded as a solution for Church's Problem (since $T_1(\alpha)(t)$ depends on $\alpha(2t)$). A fortiori, non-continuous transformations are excluded, such as T_2 defined by $T_2(\alpha) = 111\dots$ if α has infinitely many letters 1, otherwise $T_2(\alpha) = 000\dots$ (note that no finite prefix of α determines even the first bit of $T_2(\alpha)$).

The restriction to "finite-state" solutions means, in Church's words, that the desired sequence transformation should be realizable by a "circuit". This is a much stronger assumption on the admissible transformations than the dependency of the t-th output bit on the inputs bits up to time t only: One requires that the computation is realizable with a fixed finite memory (independent of t), as with the two states of memory in our example. It is remarkable that this restricted type of procedure actually suffices for solutions of Church's Problem. In this paper we work with finite-state transducers in the format of Mealy automata. Formally, a *Mealy automaton* is a structure $\mathcal{M} = (S, \Sigma, \Gamma, s_0, \delta, \tau)$ where S is the finite set of states, Σ and Γ are the input 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 $a/\tau(p,a)$. Later we shall also allow that certain transitions may not produce an output letter (but the empty word ε instead). The function δ is extended to $\delta^* : S \times \Sigma^* \to S$ by setting $\delta^*(s,\varepsilon) = s$ and $\delta^*(s,wa) = \delta(\delta^*(s,w),a)$ for $w \in \Sigma^*, a \in \Sigma$. For the input sequence $\alpha = \alpha(0)\alpha(1)\dots$, the output sequence β computed by $\mathcal{M}$ is given by $\beta(t) = \tau(\delta^*(s_0, \alpha(0)\dots\alpha(t-1)), \alpha(t))$.

Let us now make precise the specification language. 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 [4], and today it is understood that "Church's Problem" refers to S1S. In the logical context, one identifies a bit sequence α with a set P_α of nat-

ural numbers that contains the numbers t with $\alpha(t) = 1$. We use $s, t, \ldots$ as first-order variables (ranging over natural numbers, or time instances) and $X, Y, \ldots$ as second-order variables (ranging over sets of natural numbers, or bit sequences). We view the latter as predicate variables and write $X(t)$ rather than $t \in X$. In S1S, one has quantifiers $\exists, \forall$ for both kinds of variables. One can define $s < t$ by saying that t belongs to each set which contains $s + 1$ and is closed under successor. Now our example specification takes the form of the following S1S-formula $\varphi_0(X, Y)$ (where we write $\exists^\omega t \ldots$ for $\forall s \exists t(s < t \wedge \ldots)$):

$$\forall t(X(t) \to Y(t)) \ \wedge \ \neg\exists t(\neg Y(t) \wedge \neg Y(t+1)) \ \wedge \ (\exists^\omega t \neg X(t) \to \exists^\omega t \neg Y(t))$$

In general, we have S1S-specifications that speak about sequences $\alpha \in (\{0,1\}^{m_1})^\omega$ and $\beta \in (\{0,1\}^{m_2})^\omega$. Then we consider bit vectors rather than single bits, and use m_1-tuples $\overline{X}$ and m_2-tuples $\overline{Y}$ of second-order variables in place of X, Y in the specifications. Similarly we write $\overline{P_\alpha}$ for the predicate tuple associated with α. Church's Problem now asks: *Given an S1S-specification $\varphi(\overline{X}, \overline{Y})$, construct a Mealy automaton $\mathcal{M}$ with input alphabet $\Sigma = \{0,1\}^{m_1}$ and output alphabet $\Gamma = \{0,1\}^{m_2}$ such that for each input sequence $\alpha \in (\{0,1\}^{m_1})^\omega$, an output sequence $\beta \in (\{0,1\}^{m_2})^\omega$ is produced by $\mathcal{M}$ with $(\mathbb{N}, +1) \models \varphi[\overline{P_\alpha}, \overline{P_\beta}]$, or provide the answer that such an automaton does not exist.*

An alternative view to study Church's Problem is to consider a relation $R \subseteq \{0,1\}^\omega \times \{0,1\}^\omega$ as the definition of 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 $t = 0, 1, \ldots$ by A and B in alternation, and the play $(\alpha(0), \beta(0))\ (\alpha(1), \beta(1)) \ldots$ is won by player B iff the pair (α, β) belongs to R. 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 1969, Büchi and Landweber solved Church's Problem in [2]. The original proof involved a complicated construction. It took some time until more accessible proofs were available. The purpose of this tutorial is to present a construction which is made easy by a decomposition of the task into simpler modules (following [18], see also [19, 7]). The construction also gives extra information on the structure of the finite-state machines that serve as solutions.

We will show the Büchi-Landweber Theorem in four stages: In a preliminary step, the S1S-specifications are converted into automata over infinite words ("ω-automata"). Here we use, without going into details, classical results of Büchi and McNaughton that provide such a conversion ([1, 10]). We will illustrate this step by an example. Then we transform the obtained automaton into a game between the input player A and the output player B

(played essentially on the transition graph of the automaton). The task is then to decide whether B has a winning strategy and – if so – to construct a finite-state machine executing a winning strategy. The last two stages serve to obtain such a machine. First, we define its state space and transition function, and secondly we fix the output function. Only in this last step the decision about solvability of the specification will be obtained.

There is also an alternative approach to Church's Problem, developed by Rabin [16] in the framework of tree automata theory. Let us briefly sketch the idea. In the situation where both players A and B select bits, Rabin codes a strategy of player B by a labelling of the nodes of the infinite binary tree: The root has no label, the directions left and right represent the bits chosen by A, and the labels on the nodes different from the root are the bits chosen by B according to the considered strategy. When player A chooses the bits $b_0, \ldots, b_k$, he defines a path to a certain node; the label b of this node is then the next choice of player B. Note that a node labelling by bits corresponds to a subset X of the tree (containing the nodes with label 1). Now the paths through the (X-labelled) tree capture all plays that are compatible with B's strategy coded by X. One can write down a formula $\chi(X)$ in MSO-logic over the binary tree which states that the winning condition is satisfied by each path; thus $\chi(X)$ says that "X is a winning strategy". By Rabin's Tree Theorem [15] one can convert $\chi(X)$ into a Rabin tree automaton $\mathcal{A}_\chi$ (for definitions see e.g. [19]), check whether this automaton accepts some tree, and – if this is the case – construct a "regular" tree accepted by $\mathcal{A}_\chi$. This regular tree can then be interpreted as a finite-state winning strategy for player B.

In the present notes we pursue the "linear" approach in which single plays are the main objects of study; so we avoid here the infinite tree structure that captures *all* plays for a given strategy.

2 From Logic to Automata and Games

2.1 From Logic to Automata

Our 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, namely into a deterministic automaton $\mathcal{A}_\varphi$ working over ω-sequences. This 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 a compositional approach of synthesis that is guided by the structure of the formula φ.

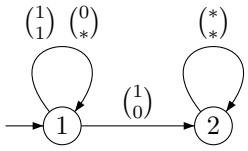
For an m_1-tuple $\overline{X}$ and an m_2-tuple $\overline{Y}$, the input alphabet of $\mathcal{A}_\varphi$ is $\{0,1\}^{m_1+m_2}$. The automaton is said to be equivalent to $\varphi(\overline{X}, \overline{Y})$ if it accepts precisely those ω-words which define tuples $(\overline{P}, \overline{Q})$ of sets such that $(\mathbb{N}, +1) \models \varphi[\overline{P}, \overline{Q}]$. In the game theoretic view explained above, one may

consider the automaton as a referee who watches the play evolving between players A and B that consists of the two sequences α and β (logically speaking: of the set tuple $(\overline{P}_\alpha, \overline{Q}_\alpha)$ built up by A and B), and who decides at infinity (by the acceptance condition) whether B has won or not.

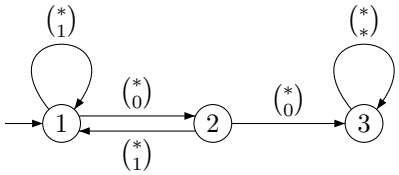
An appropriate acceptance condition is the so-called Muller condition. It is specified by a collection $\mathcal{F} = \{F_1, \ldots, F_k\}$ of state sets, and the automaton accepts an ω-word γ if the set of the states visited infinitely often in the unique infinite run on γ is one of the F_i. (The sets F_i are called *accepting loops*; indeed, if the states in F_i are visited again and again they form a strongly connected set ("loop") in the transition graph of the automaton.)

We use here two core results of the theory of ω-automata due to Büchi [1] and McNaughton [10] (see e.g. [19] or [7] for more recent expositions). They allow to translate an S1S-formula into an equivalent (non-deterministic) Büchi automaton, which is then transformed into a deterministic Muller automaton: *For each S1S-formula $\varphi(\overline{X}, \overline{Y})$ one can construct an equivalent Muller automaton $\mathcal{A}_\varphi$.* As a drawback in this result we mention that the size of $\mathcal{A}_\varphi$ cannot be bounded by an elementary function in the length n of φ (see, e.g., [7]); in other words, for no k, the k-fold iteration of the function $n \mapsto 2^n$ can serve as an upper bound for the size of $\mathcal{A}_\varphi$.

Let us illustrate the theorem for our example specification above. The formula $\forall t(\alpha(t) = 1 \to \beta(t) = 1)$ is equivalent to the Muller automaton



with accepting loop $\{1\}$ only (and where $*$ stands for an arbitrary bit). The formula $\neg\exists t\beta(t) = \beta(t+1) = 0$ is expressed by the following Muller automaton with accepting loops $\{1\}$ and $\{1, 2\}$:



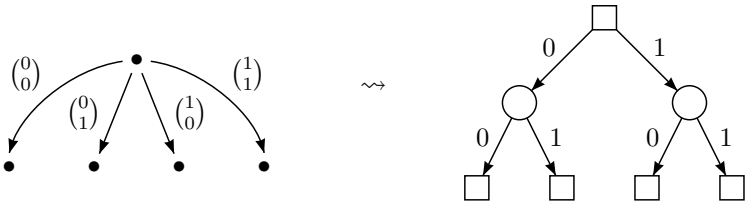
The automaton for $\exists^\omega t\alpha(t) = 0 \to \exists^\omega t\beta(t) = 0$ is best explained as follows: There are four states, denoted by the four possible bit-pairs, with say $(0, 0)$

as initial state. From each state we have, for each bit pair (b_1, b_2), a transition labelled (b_1, b_2) to the state (b_1, b_2). A set F is accepting if it satisfies the following condition: If the first component is 0 in some state of F, then the second component is 0 for some (possibly different) state of F.

It is known how to combine Muller automata for several conditions to a single Muller automaton for their conjunction. We do not present it explicitly here for our example. Rather we turn to a variant, called "finite-state game with Muller winning condition". This approach, introduced by McNaughton [11], is motivated by the view that the two components of an input letter of the Muller automaton are contributed by two players A and B who pursue antagonistic objectives: A aims at violating the condition φ and B at satisfying it.

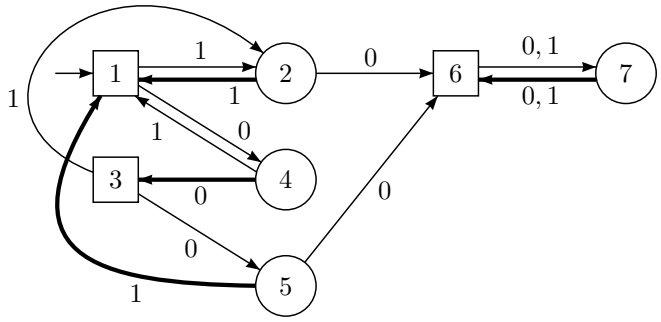
2.2 From Automata to Games

We distinguish the contribution of bits (in the general case: bit vectors) by two players A and B by introducing 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. We indicate A-states by boxes and B-states by circles. The figure below indicates how we dissolve transitions from a state in the given Muller automaton by introducing intermediate states and corresponding transitions.



Note that we keep every state of the Muller automaton as an A-state. For each A-state q and bit b, we introduce a b-labelled transition to a new state called (q, b), and from (q, b) for each bit c a c-labelled transition to the state p which was reached from q by $\binom{b}{c}$ in the original automaton. For such a state p we call c the corresponding "output bit", denoted $\mathrm{out}(q, b, p)$. (If both c-transitions from (q, b) lead to the same state p we agree that $\mathrm{out}(q, b, p) = 0$.) If the input alphabet is $\{0,1\}^{m_1}$ and the output alphabet $\{0,1\}^{m_2}$, we introduce B-states $(q, \overline{b})$ with $\overline{b} \in \{0,1\}^{m_1}$, and define $\mathrm{out}(q, \overline{b}, p)$ as a vector in $\{0,1\}^{m_2}$.

The result is a "game graph". For our example specification above, we can 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 are indeed captured by this graph. 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 the acceptance condition of a Muller automaton is thus turned into a winning condition in the associated game (Muller winning condition). Furthermore, 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 the Introduction.

In the remainder of the paper, we shall give a general construction that starts with a finite game graph equipped with a Muller winning condition, provides the decision whether player B wins, and in this case yields a finite-

state winning strategy.

We add some remarks on the step from automata to game graphs. First let us note that there is more behind this reshaping of the model than introducing the attractive idea of a game. The game theoretic view is most useful for introducing a symmetry between inputs and outputs in Church's Problem. The two players A and B represent the antagonistic aims of falsifying the specification (player A, supplying input) and satisfying it (player B, supplying output). It will turn out that either A or B has a winning strategy, an aspect which is hidden in the original formulation of Church's Problem.

Secondly, in studying plays over a given game graph, it is useful to ignore the special role occupied by the initial state. Rather we shall be interested in plays wherever they start, and we shall determine for each state which player has a winning strategy for plays starting from there.

On the other hand, we shall simplify the model in a different detail: We 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 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.

In our example, a Mealy automaton with two states was sufficient to solve Church's Problem for the specification φ_0. These two states were already present in the game graph G_{φ_0} corresponding to the Muller automaton $\mathcal{A}_{\varphi_0}$. (We took the states 1 and 3.) Given the game graph G_{φ_0}, we were able to fix the moves of player B from 1 and 3 independent of the "play history", i.e. independent of the path on which either of these states was reached. In general we shall need additional memory to define the right choice. We shall see that a finite memory suffices; so we can work with winning strategies that are implementable by finite-state machines. Such a finite-state machine $\mathcal{S}$ works on the game graph G. The states of $\mathcal{S}$ and of G should not be confused. For the solution of Church's Problem (given a logical formula φ) we have to combine the states of $\mathcal{S}$ with the states of G. We describe this in detail at the end of the next section.

3 Infinite Games and the Büchi-Landweber Theorem

A *game graph (or arena)* has the form $G = (Q, Q_A, E)$ where $Q_A \subseteq Q$ and $E \subseteq Q \times Q$ is the transition relation, satisfying $\forall q \in Q : qE \neq \varnothing$ (i.e. $\forall q \exists q' : (q, q') \in E$). This condition ensures that plays cannot end in a deadlock. (So 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; however

the results do not depend on this assumption. *We restrict to finite game graphs throughout the paper.*

A play over G from q is an infinite sequence $\rho = q_0q_1q_2\ldots$ 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. Note that the game graph is finite whereas the plays on it are infinite; thus one speaks of "finite-state infinite games".

Formally, a *game* is a pair (G, W) where $G = (Q, Q_A, E)$ is a game graph and $W \subseteq Q^\omega$ a "winning condition" for player B. Player B wins the play $\rho = q_0q_1q_2\ldots$ if $\rho \in W$, otherwise player A wins ρ. The use of such "abstract" winning conditions W is pursued in descriptive set theory, see [12]. In our algorithmic context we have to work with finitely presentable sets W. For our considerations below, we work with two finite presentations of winning conditions, either by a collection $\mathcal{F} \subseteq 2^Q$ of sets $R \subseteq Q$, or by 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.

First we introduce two winning conditions connected with a collection $\mathcal{F} \subseteq 2^Q$. The first is the *Muller winning condition*; it refers to the set $\mathrm{Inf}(\rho)$ of states visited infinitely often in a play ρ:

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

Player B wins the play ρ if $\mathrm{Inf}(\rho) \in \mathcal{F}$. With these conventions we speak of a *Muller game* $(G, \mathcal{F})$.

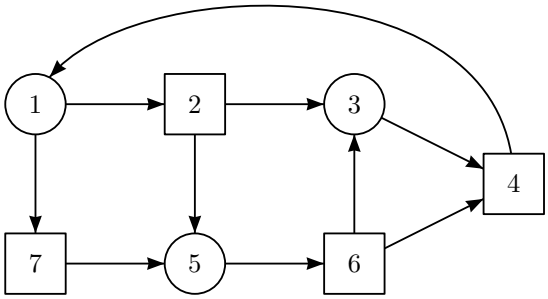
There is also a "weak" version of this winning condition, called *weak Muller condition* (or *Staiger-Wagner condition*), which refers to the visited states in a play ("occurrence set"):

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

Player B wins a play ρ according to the weak Muller condition if $\mathrm{Occ}(\rho) \in \mathcal{F}$. We speak of the *weak Muller game* $(G, \mathcal{F})$.

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 winning condition for player B is satisfied for a play ρ if some state of ρ belongs to F. We obtain an equivalent weak Muller condition if we set $\mathcal{F} = \{R \subseteq Q \mid R \cap F \neq \varnothing\}$.

The next step is to introduce the concepts of strategy, winning strategy, and winning region. Let us look at examples first, using the following game graph G_1.



The reachability game $(G_1, \{3\})$ with "goal set" $F = \{3\}$ is won by player B if the play starts in state 3. Otherwise player A can avoid this state by going from 2 to 5 and from 6 to 4. We shall say that the *winning region* of player A in this game is the set $\{1, 2, 4, 5, 6, 7\}$ and that of player B the set $\{3\}$. As a second example, consider the condition that states 2 and 7 both have to be visited again and again. Formally, this is the Muller game $(G_1, \mathcal{F})$ where $\mathcal{F}$ consists of all sets $R \supseteq \{2, 7\}$. Obviously, player B can win from any state: From 1 he proceeds to 2 and to 7 in alternation, from 5 he moves to 6, and from 3 to 4. So the winning region of A is empty in this case, and that of B the set of all states. Note that switching between the moves from 1 to 2 and from 1 to 7 means to use memory (here only one bit) when executing the strategy.

Formally, a *strategy for player B from* q is a function $f : Q^+ \to Q$, specifying for any play prefix $q_0 \dots q_k$ 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 $\rho = q_0 q_1 \dots$ from $q_0 = q$ is played according to strategy f if for each $q_i \in Q_B$ we have $q_{i+1} = f(q_0 \dots q_i)$. 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 A (resp. B) has a winning strategy from q.

For a game (G, W) with $G = (Q, Q_A, E)$, 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 (and prove) 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.

By the solution of a game (G, W), with game graph $G = (Q, Q_A, E)$ and a finitely presented winning condition W, we mean 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).

Item 2 asks for a winning strategy that has a finite presentation. Two kinds of strategies will be central in the sequel, the positional and the finite-state strategies. A strategy $f : Q^+ \to Q$ is *positional* if the value of $f(q_1 \dots q_k)$ only depends on the "current state" q_k. So a positional strategy for B can also be presented as a function $f : Q_B \to Q$, or – in graph theoretical terms – by a subset of the edge set where from Q_A-states all edges are kept but from each Q_B-state precisely one edge is chosen. For the definition of finite-state strategies, we first observe that over a finite state 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 a Mealy automaton is of the form $\mathcal{S} = (S, Q, Q, s_0, \delta, \tau)$ 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 $\delta^*(s, w)$ is the state reached by $\mathcal{S}$ from s via input word w, as defined as above in the Introduction, and τ is chosen for the player under consideration).

Now we can state the main theorem on weak Muller games and on Muller games. We include part (a) for reasons of exposition; part (b) is the Büchi-Landweber Theorem.

Theorem 3.1. (a) Weak Muller games are determined, and for a weak Muller game $(G, \mathcal{F})$, where G has n states, one can effectively determine the winning regions of the two players and construct, for each state q of G, a finite-state winning strategy from q for the respective winning player, using 2^n memory states.
(b) Muller games are determined, and for a Muller game $(G, \mathcal{F})$, where G has n states, one can effectively determine the winning regions of the two players and construct, for each state q of G, a finite-state winning strategy from q for the respective winning player, using $n! \cdot n$ memory states.

Before entering the proof, we remark that part (b) gives the desired solution of Church's Problem. For this, we proceed as in the previous section,

i.e. we transform a given S1S-formula φ to a Muller automaton $\mathcal{M}$ which is then converted to a game graph G with Muller winning condition (see Section 2). 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 now easily constructed as a product automaton from $\mathcal{M}$ and $\mathcal{S}$.

We give the complete definitions for the reader who wants to see the details. For simplicity we consider the case $\varphi(X, Y)$ where each player picks single bits only. Let $\mathcal{M}$ be the Muller automaton obtained from $\varphi(X, Y)$, say with state set Q. The game graph G derived from $\mathcal{M}$ has Q as the set of A-states and $Q \times \{0, 1\}$ as the set of B-states. Denote $Q \cup (Q \times \{0, 1\})$ by Q_0. Let $\mathcal{S} = (S, Q_0, Q_0, s_0, \delta, \tau)$ be the Mealy automaton that realizes a finite-state winning strategy for player B in the Muller game over G from q_0 (the initial state of the Muller automaton). We construct the Mealy automaton $\mathcal{A}$ solving the considered instance φ of Church's Problem as follows: $\mathcal{A}$ has the state set $Q \times S$ and the initial state (q_0, s_0). We have to specify a transition for each state (q, s) and input bit b, i.e. an output bit b' and a new state (q', s'). For this we compute the state $q^* = (q, b)$ of the game graph and the associated $\mathcal{S}$-state $s^* = \delta(s, q^*)$. The output function of $\mathcal{S}$ yields the state $q' = \tau(s, q^*)$ of G and the new memory state $s' = \delta(s^*, q')$. The output bit b' is the value $\mathrm{out}(q, b, q')$ associated to the transition from $q^* = (q, b)$ to q' (cf. Section 2.2).

The memory of the automaton $\mathcal{A}$ combines the state space of the Muller automaton $\mathcal{M}$ and that of the strategy automaton $\mathcal{S}$. It is not yet well understood how these two aspects play together in general. Our example in Sections 1 and 2 illustrates the case that in addition to the states of $\mathcal{M}$ no additional memory is necessary.

4 Reachability Games and Weak Muller Games

In this section we outline the proof of Theorem 3.1 (a). As a preparatory step we solve reachability games. The fundamental construction involved in this solution (computation of "attractor") later enters also in the solution of weak Muller games and Muller games. For this purpose, we introduce a second technique, called "game simulation". It allows to transform a given game into another one with an "easier" winning condition, namely such that the method as known from reachability games applies. We shall illustrate this approach first for weak Muller games (in this section) and then for Muller games (in the next section).

4.1 Reachability Games

Recall that a reachability game (G, F) involves the winning condition (for player B) that the play should reach somewhere a state from the set F.

Theorem 4.1. 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 positional winning strategies.

Proof. The proof follows a natural idea, namely to compute, for $i = 0, 1, \ldots$, 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). Its computation for increasing i is known from the theory of finite games (and corresponds to the well-known analysis of AND-OR-trees).

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

The inductive computation is obvious:

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

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 positional 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 $k \leq |Q|$. We 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 positional 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.)

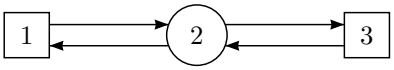
For the converse inclusion $W_B \subseteq \mathrm{Attr}_B(F)$ we show that $\mathrm{Attr}_B(F)$ exhausts the winning region W_B. For this, we show that from each state in the complement of $\mathrm{Attr}_B(F)$, player A has a winning strategy (which is again positional). It suffices to verify that from 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 $\mathrm{Attr}_B(F)$. Q.E.D.

4.2 Weak Muller Games

In a weak Muller game $(G, \mathcal{F})$, player B wins the play ρ iff $\mathrm{Occ}(\rho) \in \mathcal{F}$, i.e. the states visited during ρ form a set in $\mathcal{F}$. It is a useful exercise to verify that weak Muller games are precisely those where the winning condition can be expressed as a Boolean combination of reachability conditions.

Positional strategies do not suffice in general to win weak Muller games. As an example, consider the following game graph and the weak Muller condition given by $\mathcal{F} = \{\{1, 2, 3\}\}$ (requiring that player B should visit all states in order to win).



From vertex 2 there is no positional winning strategy: Neither the choice to move to 1 nor the choice to move to 3 will enable us 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. 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).

We shall see that this idea suffices completely for setting up the transition structure of a finite-state winning strategy. Given a weak Muller game $(G, \mathcal{F})$ with $G = (Q, Q_A, E)$ and $\mathcal{F} = \{F_1, \ldots, F_k\}$, $F_i \subseteq Q$, we define the transition structure of a Mealy automaton $\mathcal{S}$ with the power set of Q as its set of states and Q as its input alphabet. Having read the input word $q_1 \ldots q_k$, its state will be $\{q_1, \ldots, q_k\}$. So the initial state is $\varnothing$ 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 see that this memory structure suffices for winning strategies in arbitrary weak Muller games over G. What remains is to fix the output function. For this purpose we study an expanded game into which the memory contents from 2^Q are incorporated. It will turn out that – based on this extra information – the winning condition can be reformulated for the expanded game. We call this transformation of the game a "game simulation". For the new game we shall provide *positional* winning strategies, which will supply the desired output function for the strategy automaton $\mathcal{S}$.

4.3 Game Simulation

During a play ρ, the set of visited states increases weakly monotonically and finally reaches the value $\mathrm{Occ}(\rho)$ on which it stays fixed. Similarly the cardinality of the set of visited states increases until it reaches the value $|\mathrm{Occ}(\rho)|$. This observation enables us to reformulate the weak Muller winning conditon "$\mathrm{Occ}(\rho) \in \mathcal{F}$". 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 $2 \cdot |R| - 1$. Let

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

for $R \neq \varnothing$ and set $c(\varnothing) := 0$. Then the following claim is obvious:

Remark 4.2. Let ρ be a play and $R_0, R_1, R_2, \ldots$ be the value sequence of the associated appearance records. Then $\mathrm{Occ}(\rho) \in \mathcal{F}$ 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 $\rho = r_0 r_1 r_2 \ldots$ 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 the idea above, 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 for nonempty R define $c(R, r) := 2 \cdot |R|$ if $R \in \mathcal{F}$, otherwise $2 \cdot |R| - 1$ (and let $c(\varnothing, r) = 0$).

Each play $\rho = r_0 r_1 \ldots$ in G induces the play $\rho' = (\varnothing, r_0)(\{r_0\}, r_1)\ldots$ 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". In general, we say that the game (G, W) with $G = (Q, Q_A, E)$ is simulated by (G', W') with $G' = (Q', Q'_A, E')$ if there is a finite automaton $\mathcal{S} = (S, Q, s_0, \delta)$ without final states such that

- $Q' = S \times Q, Q'_A = S \times Q_A$,
- $((r, p), (s, q)) \in E'$ iff $(p, q) \in E$ and $\delta(r, p) = s$ (which means that a play $\rho = q_0 q_1 \ldots$ in G induces the play $\rho' = (s_0, q_0)(\delta(s_0, q_0), q_1)\ldots$ over G'),

- a play ρ over G belongs to W iff the corresponding play ρ' over G' belongs to W'.

If these conditions hold we write $(G, W) \leq_{\mathcal{S}} (G', W')$.

This relation has an interesting consequence when the latter game allows positional winning strategies. Namely, positional strategies over G' are easily translated into finite-state stategies over G: The automaton $\mathcal{S}$ used for the simulation realizes such a strategy when equipped with an output function that is obtained from the positional strategy over $G' = (S \times Q, S \times Q_A, E')$.

Remark 4.3. Let $\mathcal{S} = (S, Q, s_0, \delta)$ and assume $(G, W) \leq_{\mathcal{S}} (G', W')$. If there is a positional winning strategy for player B in (G', W') from (s_0, q), then player B has a finite-state winning strategy from q in (G, W). The analogous claim holds for player A.

Proof. Consider the case of player B. We extend the automaton $\mathcal{S}$ by an output function that is extracted from the winning strategy $\sigma : Q'_B \to Q'$. It suffices to define $\tau : S \times Q_B \to Q$ by $\tau(s, q) :=$ second component of $\sigma(s, q)$. Then any play ρ according to the strategy $\mathcal{S}$ belongs to W iff the corresponding play ρ' (obtained as defined via $\mathcal{S}$) belongs to W'. Since σ was assumed to be a winning strategy, so is the strategy executed by $\mathcal{S}$. The case of player A is handled analogously. Q.E.D.

We apply this remark for the concrete simulation of weak Muller games by weak parity games mentioned above. We show "positional determinacy" for weak parity games and thus – by the preceding remark – finish the proof of part (a) of Theorem 3.1, concerning weak Muller games.

Theorem 4.4. A weak parity game (G, c) is determined, and one can compute the winning regions W_A, W_B and also construct corresponding positional winning strategies for the players A and B.

It may be noted that we suppressed the initial states q when speaking about positional 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). 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 nodes form the set $Q \setminus A_k$; this 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 $A_k = \mathrm{Attr}_B(C_k)$.

In the subgame induced by $Q \setminus A_k$, 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 Section 4.1).

In both sets we can single out positional 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 to the winning regions of B and A in alternation. 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 set of states that induces the subgame 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 positional 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 $j = 0, \ldots, k$:

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

We do not give this proof in detail; it is done in analogy to the case of reachability games (Section 4.1). Q.E.D.

Returning to the solution of weak Muller games, we first note that the claim of Theorem 3.1(a) on the memory size of a finite-state winning strategy (2^n memory states over a game graph with n states) is clear from the

game simulation using the structure of appearance record. It is remarkable that this method yields a finite-state winning strategy (for either player) where the transition structure depends solely on the underlying game graph; the winning condition given by the family $\mathcal{F}$ enters only later in the definition of the output function.

5 Muller Games and Parity Games

5.1 An Example

As a preparation of the proof of the Büchi-Landweber Theorem 3.1(b), we consider a game that was introduced by Dziembowski, Jurdziński and Walukiewicz in [5]. The game is parametrized by a natural number n; we consider here the case $n = 4$.

The game graph consists of A-states $1, 2, 3, 4$ (called number-states) and B-states A, B, C, D (called letter-states). There is an edge from each A-state to each B-state and conversely.

The winning condition for B is the following, for a play ρ: *The number of letter-states occurring infinitely often in ρ has to coincide with the highest number that occurs infinitely often among the number-states in ρ.* More formally we can write $|\mathrm{Inf}(\rho) \cap \{A, B, C, D\}| = \max(\mathrm{Inf}(\rho) \cap \{1, 2, 3, 4\})$. Note that this defines a Muller game; the family $\mathcal{F}$ of accepting loops contains each set R such that $|R \cap \{A, B, C, D\}| = \max(R \cap \{1, 2, 3, 4\})$.

It is the job of player A to choose letter-states. If, for instance, player A decides after some time to stick just to the letters A and D (in some order) and not to visit B and C anymore, then player B should infinitely often pick state 2 and only finitely often the larger states 3 and 4.

From a naive point of view it is hard to imagine how player B can win this game. After a finite play prefix, nothing about the set $\mathrm{Inf}(\rho) \cap \{A, B, C, D\}$ is decided (in fact, player A has complete freedom to go for any nonempty subset of $\{A, B, C, D\}$). However, a strategy has to select one number vertex on the basis of the current finite play prefix alone.

Nevertheless, player B wins this game from each of the states, and the winning strategy illustrates again that for appropriate decisions on the future it may be sufficient to remember relevant facts from the past. We shall use a refined version of the appearance record, in which not only the visited states, but also their *order of last visits* is taken into account. In the present example, it suffices to record the list of previously visited letter-states in the order of their last visits – most recently visited states noted first. If the current (letter-) state was already visited before, then it is shifted from its previous position, say at place h in the list, to the front. The position h from which it was taken is underlined; we call it the "hit". This structure was introduced by McNaughton in [9] under the name "order-vector". Later Gurevich and Harrington suggested in their fundamental paper [8]

the name "latest appearance record" (LAR) under which the structure is known today.

Let us study an example. Suppose player A picks successively the letter-states $A, C, C, D, B, D, C, D, D, \ldots$. We note this sequence on the left, and the associated sequence of latest appearance records on the right:

Visited letter	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	$(\underline{D}CBA)$

Now assume that player A indeed sticks to the states C and D and repeats these two infinitely often. Then the states A and B will finally stay on the last two LAR-positions and not be touched anymore. Thus the hit value will be only 1 or 2 from some point onwards, and the maximal hit value visited infinitely often will be 2. In fact, if only position 1 is underlined from some point onwards, then only the same letter would be chosen from that point onwards (and not two states C and D as assumed).

We conclude that player B should always move to the number state named by the current hit value. In the scenario mentioned, this would mean to move finally only to states 1 or 2, and to 2 infinitely often. If at some point player A would decide to go to one state only, this state would be repeated at the head of the LAR and underlined; so the maximal hit value visited infinitely often would be 1 (and correct again).

We leave it to the reader to show that "to move to the number given by the current hit value" is a winning strategy of player B in the game (see also Remark 5.1 below). Since the required memory is finite and the update rule is defined in terms of the previous LAR and the current state, this is a finite-state winning strategy.

The example suggests a solution of Muller games in very close analogy to the case of weak Muller games, using the latest appearance record in place of the appearance record. We shall introduce the LAR-structure in general (i.e., we take it to cover *all* states of the game graph under consideration and not only a subset, such as the letter-states in our example). From each LAR we extract an "index". Whereas in an appearance record we referred to its cardinality, we use the hit value for a LAR. Then we introduce the "parity condition" (a variant of the weak parity condition) as new winning

condition and apply it in a game simulation. The solution of the parity game that arises in this way gives a solution of the original Muller game.

5.2 Parity Games

Given a Muller game $(G, \mathcal{F})$ with $G = (Q, Q_A, E)$ and $Q = \{1, \ldots, n\}$, we define the transition structure of a finite-state machine $\mathcal{S} = (S, Q, s_0, \delta)$. Its state set is the set of LAR's over Q, and its purpose is to realize, given a play prefix $i_1 \ldots i_k \in Q^*$, the computation of the corresponding LAR. Formally, an LAR is a pair $((j_1 \ldots j_r), h)$ where the j_i are pairwise distinct states from Q and $0 \leq h \leq r$. The initial state is $s_0 = ((\,), 0)$ (empty list and hit 0). The transition function $\delta : S \times Q \to S$ realizes the update of the LAR as indicated in the example above: We set $\delta(((i_1 \ldots i_r), h), i) = ((ii_1 \ldots i_r), 0)$ if i does not occur in $(i_1 \ldots i_r)$. Otherwise, if $i = i_k$ we cancel i from $(i_1 \ldots i_r)$ to obtain $(j_1 \ldots j_{r-1})$ and set $\delta(((i_1 \ldots i_r), h), i) = ((ij_1 \ldots j_{r-1}), k)$.

An essential ingredient of a LAR $((i_1 \ldots i_r), h)$ is the *hit set* $\{i_1, \ldots, i_h\}$ of states listed up to and including the hit position h. Consider a play ρ over Q and the associated sequence of LAR's, denoted ρ'. If h is the maximal hit assumed infinitely often in ρ', we may pick a position 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'.

Remark 5.1. Let ρ be a sequence over Q and ρ' be the associated sequence of LAR's. The set $\mathrm{Inf}(\rho)$ coincides with the hit set H for the maximal hit h occurring infinitely often in ρ'.

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 each state in $\mathrm{Inf}(\rho)$, these states will stay at the head of the LAR's (in various orders), and the hit values will be $\leq k := |\mathrm{Inf}(\rho)|$. It remains to show that the hit value in ρ' reaches k again and again (so that k is the maximal hit occurring infnitely 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 $\mathrm{Inf}(\rho)$. Q.E.D.

Using the remark, we can reformulate the Muller winning condition for the play ρ: *The hit set for the highest 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, for $h > 0$,

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

and let $c(((i_1 \dots i_r), 0)) = 0$.

Then the Muller condition $\mathrm{Inf}(\rho) \in \mathcal{F}$ is satisfied iff the maximal color occurring infinitely often in $c(\rho'(0))c(\rho'(1))\dots$ is even. This is a "parity condition" (as introduced by Mostowski [13] and Emerson and Jutla [6]). The only difference to the weak parity condition is the reference to colors occurring infinitely often rather than 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:

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

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

Similar 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): We use the finite-state machine $\mathcal{S}$ introduced before that transforms a given play ρ over G into the corresponding sequence ρ' of LAR's (realized in the states visited by $\mathcal{S}$), and we use the coloring c defined above. The game graph G' is fixed as in Section 4.3 above, using the new machine $\mathcal{S}$. We obtain the game simulation $(G, \mathcal{F}) \leq_{\mathcal{S}} (G', c)$ where (G', c) is a parity game.

Remark 5.2. There is a variant of $\mathcal{S}$ in which some of the states are spared. We cancel the initial LAR's (corresponding to hit value 0), starting (over states $1, \dots, n$) with the LAR $((1 \dots n), 1)$ rather than $(( ), 0)$, and keeping the update rule as before. With this change, one cannot distinguish between first and repeated visits of states, but clearly this loss of information is inessential for the satisfaction of the winning condition. The number of states of the reduced machine is then $n! \cdot n$ over a graph with n states.

One can use $\mathcal{S}$ as the transition structure of automata realizing winning strategies in the Muller game $(G, \mathcal{F})$. In order to provide also the output function, we have to solve parity games, again by positional winning strategies.

Theorem 5.3. A parity game (G, c) is determined, and one can compute the winning regions W_A, W_B and also construct corresponding positional 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 $Q \setminus A_0$ induces a subgame. The induction hypothesis ensures a partition of $Q \setminus A_0$ into the winning regions U_A, U_B of the two players (with corresponding positional winning strategies) in this subgame.

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 $W_B = U_B \cup \mathrm{Attr}_B(\{q\})$ and $W_A = U_A$, applying the positional strategies of the induction hypothesis over U_A, U_B, the attractor strategy over $\mathrm{Attr}_B(\{q\})$, and (if $q \in Q_B$) the choice of the next state from q according to Case 1. For the first claim, note that a play 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 (by choice 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 second claim $W_A = U_A$ is now clear by induction hypothesis.

We turn to Case 2. In this case we know that $q \in \mathrm{Attr}_A(U_A)$ and 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 subgame induced by $Q \setminus A_1$. We obtain a partition of this domain into winning regions V_A, V_B for A and B, with corresponding positional winning strategies. Now it is easy to verify $W_B = V_B$ and $W_A = V_A \cup A_1$, with positional winning strategies again provided by the induction hypothesis and the attractor strategy over A_1.

Finally we note that the inductive construction can be turned to a recursive procedure which produces, given G and the coloring c, the desired winning regions and positional strategies. Q.E.D.

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.

As mentioned above, Theorem 5.3 on positional determinacy of parity games completes the solution of Church's Problem. The claim on the number of states of a finite-state winning strategy ($n! \cdot n$ memory states over a graph with n states) is clear from Remark 5.2. As shown in [5], the factorial function also supplies a lower bound on the memory size of winning strategies in Muller games.

It is worth noting that the claim on positional determinacy of parity games also holds for infinite game graphs (however, without a statement on computability of winning strategies). This "infinite version" of the theorem can be applied for the complementation of automata over infinite trees (see [19]).

6 Conclusion

Let us recall the three major steps for a solution of Church's Problem: First we relied on a translation from the logic S1S to Muller automata, which were then changed into game graphs with Muller winning condition. From Muller games we constructed parity games via the LAR structure; and finally we presented a solution of parity games. All three steps are nontrivial. As mentioned, the first step involves a non-elementary blow-up (from length of formula to size of automaton). For each of the other two steps, an exponential time procedure was presented; a direct construction is possible, however, resulting in a single exponential altogether (see [20]). On the other hand, our two-step approach showed that finite-state winning strategies for a Muller game over a graph G can be constructed with a transition structure that depends on G alone, and that only for the output function the winning condition has to be invoked.

Church's Problem and its solution were the starting point for a highly active area of research in computer science, first restricted to pure automata theory, but in the last 20 years with a great influence in algorithmic verification and program synthesis. A problem in current research is to find classes of infinite game graphs over which games with MSO-definable winning conditions can still be solved algorithmically. Some results (on so-called pushdown graphs) are mentioned in [7]. Another direction is to modify or to generalize the specification language in Church's Problem (see e.g. [17]). In a wider context, more general models of games are studied, for instance "concurrent games" (where the two players move simultaneously), "timed games" (generalizing the model of timed automata), stochastic games (in which random moves enter), and multiplayer games.

7 Acknowledgment

Thanks are due to Erich Grädel, Wong Karianto, Detlef Kähler, Christof Löding, and Michaela Slaats for their helpful comments on a previous version of this paper.

References

[1] J.R. Büchi. On a decision method in restricted second order arithmetic, in *Proc. 1960 International Congress on Logic, Methodology and Philosophy of Science*, E. Nagel at al., eds, Stanford University Press 1962, pp. 1-11.

[2] J.R. Büchi, L.H. Landweber. Solving sequential conditions by finite-state strategies, *Trans. Amer. Math. Soc.* 138 (1969), 367-378.

[3] A. Church. Applications of recursive arithmetic to the problem of circuit synthesis, in *Summaries of the Summer Institute of Symbolic Logic*, Cornell Univ., Ithaca, N.Y. 1957, Volume I, pp. 3-50.

[4] A. Church. Logic, arithmetic, and automata, in *Proc. Int. Congr. Math. 1962*, Inst. Mittag-Leffler, Djursholm, Sweden, 1963, pp. 23-35.

[5] S. Dziembowski, M. Jurdziński, I. Walukiewicz. How much memory is needed to win infinite games?, in *Proc. 12th LICS*, IEEE Comp. Soc. 1997, pp. 99-110.

[6] E.A. Emerson, C.S. Jutla. Tree automata, mu-calculus, and determinacy, in *Proc. 32nd FoCS 1991*, IEEE Comp. Soc. Press 1991, pp. 368-377.

[7] E. Grädel, W. Thomas, Th. Wilke (Eds). *Automata, Logics, and Infinite Games*, Springer LNCS 2500 (2002).

[8] Y. Gurevich, L. Harrington. Trees, automata, and games, in *Proc. 14th STOC*, ACM Press 1982, 60-65.

[9] R. McNaughton. Finite-state infinite games, Project MAC Rep., MIT, Sept. 1965.

[10] R. McNaughton. Testing and generating infinite sequences by a finite automaton, *Inf. Contr.* 9 (1966), 521-530.

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

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

[13] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata, in *Computation Theory*, Springer LNCS 208 (1984), 157-168.

[14] D. Perrin, J.E. Pin. *Infinite Words*, Elsevier, Amsterdam 2004.

[15] M.O. Rabin. Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* 141 (1969), 1-35.

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

[17] A. Rabinovich, W. Thomas. Logical refinements of Church's Problem, in *Proc. CSL 2007*, Springer LNCS 4646 (2007), 69-83.

[18] W. Thomas. On the synthesis of strategies in infinite games, in *Proc. STACS 1995*, Springer LNCS 900 (1995), 1-13.

[19] W. Thomas. Languages, automata, and logic, in *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, eds.), Vol 3. Springer 1997, pp. 389-455.

[20] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees, *Theor. Comput. Sci.* 200 (1998), 135-183.