Church Synthesis Problem with Parameters

Alexander Rabinovich

Dept. of CS, Tel Aviv Univ.
rabinoa@post.tau.ac.il

Abstract. The following problem is known as the Church Synthesis problem:
Input: an *MLO* formula $\psi(X, Y)$.
Task: Check whether there is an operator $Y = F(X)$ such that

$$Nat \models \forall X \psi(X, F(X)) \tag{1}$$

and if so, construct this operator.
Büchi and Landweber proved that the Church synthesis problem is decidable; moreover, they proved that if there is an operator F which satisfies (1), then (1) can be satisfied by the operator defined by a finite state automaton. We investigate a parameterized version of the Church synthesis problem. In this version ψ might contain as a parameter a unary predicate P. We show that the Church synthesis problem for P is computable if and only if the monadic theory of $\langle Nat, <, P\rangle$ is decidable. We also show that the Büchi-Landweber theorem can be extended only to ultimately periodic parameters.

1 Introduction

Two fundamental results of classical automata theory are decidability of the monadic second-order logic of order (MLO) over $\omega = (Nat, <)$ and computability of the Church synthesis problem. These results have provided the underlying mathematical framework for the development of formalisms for the description of interactive systems and their desired properties, the algorithmic verification and the automatic synthesis of correct implementations from logical specifications, and advanced algorithmic techniques that are now embodied in industrial tools for verification and validation.

Büchi [Bu60] proved that the monadic theory of $\omega = \langle Nat, <\rangle$ is decidable. Even before the decidability of the monadic theory of ω has been proved, it was shown that the expansions of ω by "interesting" functions have undecidable monadic theory. In particular, the monadic theory of $\langle Nat, <, +\rangle$ and the monadic theory of $\langle Nat, <, \lambda x.2 \times x\rangle$ are undecidable [Rob58, Trak61]. Therefore, most efforts to find decidable expansions of ω deal with expansions of ω by monadic predicates.

Elgot and Rabin [ER66] found many interesting predicates $\mathbf{P}$ for which *MLO* over $\langle Nat, <, \mathbf{P}\rangle$ is decidable. Among these predicates are the set of factorial

Z. Ésik (Ed.): CSL 2006, LNCS 4207, pp. 546–561, 2006.
© Springer-Verlag Berlin Heidelberg 2006

numbers $\{n! \ : \ n \in Nat\}$, the sets of k-th powers $\{n^k \ : \ n \in Nat\}$ and the sets $\{k^n \ : \ n \in Nat\}$ (for $k \in Nat$).

The Elgot and Rabin method has been generalized and sharpened over the years and their results were extended to a variety of unary predicates (see e.g., [Ch69, Th75, Sem84, CT02]). In [Rab05] we provided necessary and sufficient conditions for the decidability of monadic (second-order) theory of expansions of the linear order of the naturals ω by unary predicates.

Let Spec be a specification language and Pr be an implementation language. The synthesis problem for these languages is stated as follows: find whether for a given specification $S(I, O) \in$SPEC there is a program $\mathcal{P}$ which implements it, i.e., $\forall I(S(I, \mathcal{P}(I))$.

The specification language for the Church Synthesis problem is the Monadic second-order Logic of Order. An *MLO* formula $\varphi(X, Y)$ specifies a binary relation on subsets of *Nat*. Note that every subset P of *Nat* is associated with its characteristic ω-string u_P (where $u_P(i) = 1$ if $i \in P$ and otherwise $u_P(i) = 0$). Hence, $\varphi(X, Y)$ can be considered as a specification of a binary relation on ω-strings.

As implementations, Church considers functions from the set $\{0, 1\}^\omega$ of ω-strings over $\{0, 1\}$ to $\{0, 1\}^\omega$. Such functions are called operators. A machine that computes an operator at every moment $t \in Nat$ reads an input symbol $X(t) \in \{0, 1\}$ and produces an output symbol $Y(t) \in \{0, 1\}$. Hence, the output $Y(t)$ produced at t depends only on inputs symbols $X(0), X(1), \ldots, X(t)$. Such operators are called *causal* operators (C-operators); if the output $Y(t)$ produced at t depends only on inputs symbols $X(0), X(1), \ldots, X(t-1)$, the corresponding operator is called *strongly causal* (SC-operator). The sets of recursive causal and strongly causal operators are defined naturally; a C-or a SC-operator is a *finite state* operator if it is computable by a finite state automaton (for precise definitions, see Subsection 2.3).

The following problem is known as the Church Synthesis problem.

Church Synthesis problem
Input: an *MLO* formula $\psi(X, Y)$.
Task: Check whether there is a C-operator F such that
$Nat \models \forall X \psi(X, F(X))$ and if so, construct this operator.

The Church Synthesis problem is much more difficult than the decidability problem for *MLO* over ω. Büchi and Landweber [BL69] proved that the Church synthesis problem is computable. Their main theorem is stated as follows:

Theorem 1. *For every MLO formula $\psi(X, Y)$ either there is a finite state C-operator F such that $Nat \models \forall X \psi(X, F(X))$ or there is a finite state SC-operator G such that $Nat \models \forall Y \neg\psi(G(Y), Y)$. Moreover, it is decidable which of these cases holds and a corresponding operator is computable from ψ.*

In this paper we consider natural generalizations of the Church Synthesis Problem over expansions of ω by monadic predicates, i.e., over the structures $\langle Nat, <, \mathbf{P}\rangle$.

For example, let $\mathbf{Fac} = \{n! \ : \ n \in Nat\}$ be the set of factorial numbers, and let $\varphi(X, Y, \mathbf{Fac})$ be a formula which specifies that $t \in Y$ iff $t \in \mathbf{Fac}$ and $(t' \in X) \leftrightarrow (t' \in \mathbf{Fac})$ for al all $t' \leq t$. It is easy to observe that there is no finite state C-operator F such that $\forall X \varphi(X, F(X), \mathbf{Fac})$. However, there is a recursive C-operator H such that $\forall X \varphi(X, H(X), \mathbf{Fac})$. It is also easy to construct a finite state C-operator $G(X, Z)$ such that $\forall X \varphi(X, G(X, \mathbf{Fac}), \mathbf{Fac})$. It was surprising for us to discover that it is decidable whether for a formula $\psi(X, Y, \mathbf{Fac})$ there is a C-operator F such that $\forall X \varphi(X, F(X), \mathbf{Fac})$ and if such an operator exists, then it is recursive and computable from ψ.

Here is the summary of our results. We investigate a parameterized version of the Church synthesis problem. In this version ψ might contain as a parameter a unary predicate P. Below five synthesis problems with a parameter $\mathbf{P} \subseteq Nat$ are stated.

Synthesis Problems for $\mathbf{P} \subseteq Nat$

Input: an *MLO* formula $\psi(X, Y, P)$.

Problem 1: Check whether there is a C-operator $Y = F(X, P)$ such that $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$ and if there is such a recursive operator - construct it.

Problem 2: Check whether there is a recursive C-operator $Y = F(X, P)$ such that $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$ and if so - construct this operator.

Problem 3: Check whether there is a recursive C-operator $Y = F(X)$ such that $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ and if so - construct this operator.

Problem 4: Replace "recursive" by "finite state" in *Problem 2.*

Problem 5: Replace "recursive" by "finite state" in *Problem 3.*

We show

Theorem 2. *Let* $\mathbf{P}$ *be a subset of Nat. The following conditions are equivalent:*

1. *Problem 1 for* $\mathbf{P}$ *is computable.*
2. *Problem 2 for* $\mathbf{P}$ *is computable.*
3. *Problem 3 for* $\mathbf{P}$ *is computable.*
4. *The monadic theory of* $\langle Nat, <, \mathbf{P} \rangle$ *is decidable.*
5. *For every MLO formula* $\psi(X, Y, P)$ *either there is a recursive C-operator* F *such that* $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ *or there is a recursive SC-operator* G *such that* $Nat \models \forall Y \neg \psi(G(Y), Y, \mathbf{P})$. *Moreover, it is decidable which of these cases holds and the (description of the) corresponding operator is computable from* ψ.

The difficult part of this theorem is the implication (4)⇒(5). In [BL69] a reduction of the Church synthesis problem to infinite two-player games on finite graphs was provided. Our proof is based on a reduction of the Church synthesis problem with parameters to infinite two-player games on infinite graphs. The main part of the proof shows that the reduction is "uniform" in the parameters and the corresponding infinite graph can be interpreted in $(Nat, <, \mathbf{P})$. Lemma 13 from [Rab05] also plays a key role in this proof.

The trivial examples of predicates with decidable monadic theory are ultimately periodic predicates. Recall that a predicate $\mathbf{P}$ is ultimately periodic if there is $p, d \in Nat$ such that $(n \in \mathbf{P} \leftrightarrow n + p \in \mathbf{P})$ for all $n > d$. Ultimately periodic predicates are *MLO* definable. Hence, for these predicates computability of Problems 1-5 can be derived from Theorem 1.

We prove that the Büchi-Landweber theorem can be extended only to ultimately periodic parameters.

Theorem 3. *Let* $\mathbf{P}$ *be a subset of Nat. The following conditions are equivalent and imply computability of Problem 4:*

1. $\mathbf{P}$ *is ultimately periodic.*
2. *For every MLO formula* $\psi(X, Y, P)$ *either there is a finite state C-operator* F *such that* $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$ *or there is a finite state SC-operator* G *such that* $Nat \models \forall Y \neg\psi(G(Y, \mathbf{P}), Y, \mathbf{P})$.

The paper is organized as follows. In Section 2 notations are fixed and standard definitions and facts about automata and logic are recalled. In Section 3 parity games and their connection to the synthesis problems are discussed. Büchi and Landweber [BL69] used the determinacy of such games as one of the main tools to prove computability of the Church synthesis problem and derive Theorem 1. We prove here that the question about the existence of a C-operator F such that $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$ can be effectively reduced to the decidability of monadic theory of $\langle Nat, <, \mathbf{P}\rangle$.

In Section 4 a proof of Theorem 2 is provided. In Section 5 finite state synthesis problems with parameters are considered and Theorem 3 is proved. Finally, in Section 6 some open problems are stated and related works are discussed.

2 Preliminaries

2.1 Notations and Terminology

We use $k,\ l,\ m,\ n,\ i$ for natural numbers; Nat for the set of natural numbers and capital bold letters $\mathbf{P},\ \mathbf{S},\ \mathbf{R}$ for subsets of Nat. We identify subsets of a set A and the corresponding unary (monadic) predicates on A.

The set of all (respectively, non-empty) finite strings over an alphabet Σ is denoted by Σ^* (respectively, by Σ^+). The set of ω-strings over Σ is denoted by Σ^ω.

Let $a_0 \dots a_k \dots$ and $b_0 \dots b_k \dots$ be ω-strings. We say that these ω-strings coincide on an interval $[i, j]$ if $a_k = b_k$ for $i \le k \le j$. A function F from Σ_1^ω to Σ_2^ω will be called an operator of type $\Sigma_1 \to \Sigma_2$. An operator F is called *causal* (respectively, *strongly causal*) operator, if $F(X)$ and $F(Y)$ coincide on an interval $[0, t]$, whenever X and Y coincide on $[0, t]$ (respectively, on $[0, t)$). We will refer to causal (respectively, strongly causal) operators as C-operators (respectively, SC-operators).

Every SC-operator F of type $\Sigma \to \Sigma$ has a unique fixed point, i.e., there is a unique $X \in \Sigma^\omega$ such that $X = F(X)$.

Let $G : \Sigma^\omega \to \Delta^\omega$ be an operator. In the case Σ is the Cartesian product $\Sigma_1 \times \Sigma_2$ we will identify F with the corresponding operator $F : \Sigma_1^\omega \times \Sigma_2^\omega \to \Delta^\omega$. An operator $F : \Sigma_1^\omega \times \Sigma_2^\omega \to \Delta^\omega$ is said to be SC-operator (C-operator) if G is SC-operator (respectively, C-operator).

There exists a one-one correspondence between the set of all ω-strings over the alphabet $\{0,1\}^n$ and the set of all n-tuples $\langle \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ of unary predicates over the set of natural numbers. With an n-tuple $\langle \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ of unary predicates over Nat, we associate the ω-string $a_0 a_1 \ldots a_k \ldots$ over alphabet $\{0,1\}^n$ defined by $a_k =_{def} \langle b_1^k, \ldots b_n^k \rangle$ where b_i^k is 1 if $\mathbf{P}_i(k)$ holds and b_i^k is 0 otherwise. Let $Q = \{q_1, \ldots, q_m\}$ be a finite set of state. There is a natural one-one correspondence between subset of $Q \times Nat$ and the set of m-tuples of unary predicates over Nat: with $U \subseteq Q \times Nat$ we associate the m-tuple $\langle \mathbf{P}_1, \ldots, \mathbf{P}_m \rangle$ defined as $i \in \mathbf{P}_j$ iff $U(q_j, i)$ (for $i \in Nat$ and $j \leq m$).

Similarly, there is a one-one correspondence between the set of all strings of length m over the alphabet $\{0,1\}^n$ and the set of all n-tuples $\langle \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ of unary predicates over the set $\{0, \ldots, m-1\}$.

A linearly ordered set will be called a chain. A chain with n monadic predicates over its domain will be called an n-labelled chain; whenever n is clear from the context, n-labelled chains will be called labelled chains.

We will sometimes identify an n-labelled chain $M = \langle Nat, <, \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ with the ω-string over the alphabet $\{0,1\}^n$ which corresponds to the n-tuple $\langle \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$; this ω-string will be called the *characteristic* ω-string (or ω-word) of M. Similarly, we will identify finite n-labelled chains with corresponding strings over $\{0,1\}^n$.

2.2 Monadic Second-Order Logic and Monadic Logic of Order

Let σ be a relational signature. Atomic formulas of the monadic second-order logic over σ are $R(t_1, ..., t_n)$, $t_1 = t_2$, and $t_1 \in X$ where $t_1, \ldots, t_n$ are individual variables, $R \in \sigma$ is an n-are relational symbol, and X is a set variable. Formulas are obtained from atomic formulas by conjunction, negation, and quantification $\exists t$ and $\exists X$ for t an individual and X a set variable. The satisfaction relation $M, \tau_1, \ldots \tau_k; \mathbf{S}_1, \ldots, \mathbf{S}_m \models \varphi(t_1, \ldots, t_k; X_1, \ldots, X_m)$ is defined as usual with the understanding that set variables range over subsets of M.

We use standard abbreviations, e.g., we write $X \subseteq X'$ for $\forall t.\ X(t) \to X'(t)$; we write $X = X'$ for $\forall t.\ X(t) \leftrightarrow X'(t)$; symbols "$\exists^{\leq 1}$" and "$\exists!$" stands for "there is at most one" and "there is a unique".

If a signature σ contains one binary predicate $<$ which is interpreted as a linear order, and all other predicates are unary, the monadic second-order logic for this signature is called Monadic Logic of Order (MLO). The formulas of MLO are interpreted over labelled chains.

The *monadic theory* of a labelled chain M is the set of all MLO sentences which hold in M.

We will deal with the expansions of ω by monadic predicates, i.e., with the structures of the form $M = \langle Nat, <, \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$. We say that a chain $M = \langle Nat, <, \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ is *recursive* if all $\mathbf{P}_i$ are recursive subsets of Nat.

An ω-language L is said to be *defined* by an *MLO* formula $\psi(X_1, \ldots, X_n)$ if the following condition holds: an ω string is in L iff the corresponding n-tuple of unary predicates satisfies ψ.

2.3 Automata

A *deterministic transition system* D is a tuple $\langle \mathcal{Q}, \Sigma, \delta, q_{init} \rangle$, consisting of a set $\mathcal{Q}$ of *states*, an *alphabet* Σ, a *transition function* $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ and *initial state* $q_{init} \in \mathcal{Q}$. The transition function is extended as usual to a function from $\mathcal{Q} \times \Sigma^*$ to $\mathcal{Q}$ which will be also denoted by δ. The function $\delta_{init} : \Sigma^* \to \mathcal{Q}$ is defined as $\delta_{init}(\pi) = \delta(q_{init}, \pi)$. A transition systems is finite if $\mathcal{Q}$ and Σ are finite. D is recursive if (1) the sets of states and the alphabet are at most countable and (2) there are enumerations of the sets of states $\mathcal{Q} = \{q_i : i \in Nat\}$ and the alphabet $\Sigma = \{a_i : i \in Nat\}$ such that the function $\lambda i \lambda j . \delta(q_i, a_j)$ is recursive.

A *finite deterministic automaton* $\mathcal{A}$ is a tuple $\langle \mathcal{Q}, \Sigma, \delta, q_{init}, F \rangle$, where $\langle \mathcal{Q}, \Sigma, \delta, q_{init} \rangle$ is a finite deterministic transition system and F is a subset of $\mathcal{Q}$. A string $\pi \in \Sigma^*$ is accepted by $\mathcal{A}$ if $\delta_{init}(\pi) \in F$. The language accepted (or defined) by $\mathcal{A}$ is the set of string accepted by $\mathcal{A}$.

A *Mealey automaton* is a tuple $\langle \mathcal{Q}, \Sigma, \delta, q_{init}, \Delta, out \rangle$, where $\langle \mathcal{Q}, \Sigma, \delta, q_{init} \rangle$ is a deterministic transition system, Δ is an alphabet and $out : \mathcal{Q} \to \Delta$ is an output function. With a Mealey automaton $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta, q_{init}, \Delta, out \rangle$ we associate a function $h_{\mathcal{A}} : \Sigma^* \to \Delta$ and an operator $F_{\mathcal{A}} : \Sigma^\omega \to \Delta^\omega$ defined as follows:

$$h_{\mathcal{A}}(a_0 \ldots a_{i-1}) = out(\delta_{init}(a_0 \ldots a_{i-1}))$$

$$F_{\mathcal{A}}(a_0 \ldots a_i \ldots) = b_0 \ldots b_i \ldots \text{ iff } b_i = h_{\mathcal{A}}(a_0 \ldots a_{i-1}))$$

It is easy to see that an operator is strongly causal (SC-operator) iff it is definable by a Mealey automaton. We say that a SC-operator $F : \Sigma^\omega \to \Delta^\omega$ is finite state (respectively, recursive) iff it is definable by a finite state (respectively, by a recursive) Mealey automaton.

A (deterministic) *parity automaton* $\mathcal{A}$ is a finite Mealey automaton $\langle \mathcal{Q}, \Sigma, \to, \delta, q_{init}, \Delta, col \rangle$, where the output alphabet Δ is a (finite) subset of *Nat*; the output function *col* is usually named coloring function.

With an ω-string $a_0 a_1 \ldots a_i \cdots \in \Sigma^\omega$ we associate the ω-sequence $\delta_{init}(a_0)\delta_{init}(a_1)\ldots\delta_{init}(q_i)\ldots$ of states and the set $\mathtt{Inf}$ of all $q \in \mathcal{Q}$ that appear infinitely many times in this sequence. An ω-string is accepted by $\mathcal{A}$ if the minimal element of the set $\{col(q) : q \in \mathtt{Inf}\}$ is even. The ω-language *accepted* (or defined) by $\mathcal{A}$ is the set of all ω-strings accepted by $\mathcal{A}$.

Sometimes the alphabet Σ of $\mathcal{A}$ will be the Cartesian product $\Sigma_1 \times \Sigma_2 \times \Sigma_3$ of other alphabets. In this case we say that $\mathcal{A}$ defines a relation $R_{\mathcal{A}} \subseteq \Sigma_1^\omega \times \Sigma_2^\omega \times \Sigma_3^\omega$; a triplet $\langle a, b, c \rangle$ of ω-strings is in $R_{\mathcal{A}}$ iff the ω string $(a_0, b_0, c_0)(a_1, b_1, c_1)\ldots(a_i, b_i, c_i)\ldots$ is accepted by $\mathcal{A}$.

Here is the classical theorem due to Büchi, Elgot and Trakhtenbrot.

Theorem 4. *1. A language is accepted by a finite deterministic automaton iff it is definable by an MLO formula.*

2. An ω-language is accepted by a parity automaton iff it is definable by an MLO formula.

3. *Moreover, there is an algorithm which for every formula $\varphi(X_1, \ldots, X_m)$ computes an equivalent deterministic automaton $\mathcal{A}$ i.e., the language definable by φ is accepted by $\mathcal{A}$. There is an algorithm which for every deterministic automaton $\mathcal{A}$ computes an equivalent MLO formula. Similarly, there are translation algorithms between formulas and deterministic parity automata.*

A *Moore automaton* is a tuple $\langle \mathcal{Q}, \Sigma, \delta, q_{init}, \Delta, out \rangle$, where $\langle \mathcal{Q}, \Sigma, \delta, q_{init} \rangle$ is a deterministic transition system, Δ is an alphabet and $out : \mathcal{Q} \times \Sigma \to \Delta$ is an output function.

With a Moore automaton $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta, q_{init}, \Delta, out \rangle$ we associate a function $h_{\mathcal{A}} : \Sigma^+ \to \Delta$ and an operator $F_{\mathcal{A}} : \Sigma^\omega \to \Delta^\omega$ defined as follows:

$$h_{\mathcal{A}}(a_0 \ldots, a_i) = out(\delta_{init}(a_0 \ldots a_{i-1}), a_i)$$

$$F_{\mathcal{A}}(a_0 \ldots a_i \ldots) = b_0 \ldots b_i \ldots \text{ iff } b_i = h_{\mathcal{A}}(a_0 \ldots, a_i)$$

It is easy to see that an operator is causal (C-operator) iff it is definable by a Moore automaton.

We say that a C-operator $F : \Sigma^\omega \to \Delta^\omega$ is finite state (respectively, recursive) iff it is definable by a finite state (respectively, by a recursive) Moore automaton.

3 Parity Games and the Synthesis Problem

In the next subsection we provide standard definitions and facts about infinite two-player perfect information games. In [BL69] a reduction of the Church synthesis problem to infinite two-player games on finite graphs was provided. In subsection 3.2 we provide a reduction of the Church synthesis problem with parameters to infinite two-player games on infinite graphs; this reduction is "uniform" in the parameters. The main technical results needed for the proof of Theorem 2 are Lemmas 8 and 9 which roughly speaking state that and the corresponding infinite graph can be interpreted in $(Nat, <, \mathbf{P})$.

3.1 Parity Games

We consider here two-player perfect information games played on graphs in which each player chooses in turn a vertex adjacent to a current vertex. The presentation is based on [PP04].

A (directed) bipartite graph $G = (V_1, V_2, E)$ is called a *game arena* if the outdegree of every vertex is at least one. If G is an arena, a game on G is defined by an initial node $v_{init} \in V_1$ and a set of winning ω-paths $\mathcal{F}$ from this node.

Player I plays on vertices in V_1 and Player II on vertices in V_2. A play from a node v is an infinite path in G which starts at v. Player I wins if the play belongs to $\mathcal{F}$.

A strategy f for Player I (Player II) is a function which assigns to every path of even (odd) length a node adjacent to the last node of the path. A play $v_{init} v_2 v_3 \ldots$ is played according to a strategy f_1 of Player I (strategy f_2 of Player II) if for every prefix $\pi = v_{init} v_2 \ldots v_n$ of even (odd) length $v_{n+1} = f_1(\pi)$ (respectively, $v_{n+1} = f_2(\pi)$). A strategy is winning for Player I (respectively, for

Player II) if all the plays played according to this strategy are in $\mathcal{F}$ (respectively, in the complement of $\mathcal{F}$). A strategy is *memoryless* if it depends only on the last nodes in the path.

Parity games are games on graphs in which the set of winning paths are defined by parity conditions. More precisely, let $G = (V_1, V_2, E)$ be a game graph and let $c : V_1 \cup V_2 \to \{0, 1, \dots m\}$ be a coloring.

Let $\rho = v_1 v_2 \dots$ be a play. With such a play ρ, we associate the set of colors C_ρ that appear infinitely many times in the ω-sequence $col(v_1)col(v_2)\dots$; a play ρ is winning if the minimal element of C_ρ is odd. The following theorem [EJ91, GTW02, PP04] is fundamental:

Theorem 5. *In a parity game, one of the players has a memoryless winning strategy.*

3.2 Games and the Church Synthesis Problem

Let $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta_\mathcal{A}, q_{init}, col \rangle$ be a deterministic parity automaton over the alphabet $\Sigma = \{0,1\} \times \{0,1\} \times \{0,1\}$, let $R_\mathcal{A} \subseteq \{0,1\}^\omega \times \{0,1\}^\omega \times \{0,1\}^\omega$ be the relation definable by $\mathcal{A}$ and let $\mathbf{P}$ be a subset of Nat. We will define a parity game $G_{\mathcal{A},\mathbf{P}}$ such that

1. Player I has a winning strategy in $G_{\mathcal{A},\mathbf{P}}$ iff there is a SC-operator $G : \{0,1\}^\omega \to \{0,1\}^\omega$ such that $R_\mathcal{A}(G(Y), Y, \mathbf{P})$ holds for every Y.
2. Player II has a winning strategy in $G_{\mathcal{A},\mathbf{P}}$ iff there is a C-operator $F : \{0,1\}^\omega \to \{0,1\}^\omega$ such that $\neg R_\mathcal{A}(X, F(X), \mathbf{P})$ holds for every X.

The arena $G(V_1, V_2, E)$ of $G_{\mathcal{A},\mathbf{P}}$ is defined as follows:

1. $V_1 = \mathcal{Q} \times Nat$ and $V_2 = \mathcal{Q} \times \{0,1\} \times Nat$.
2. From $\langle q, n \rangle \in V_1$ there exit two edges; one to $\langle q, 0, n \rangle \in V_2$, and the second to $\langle q, 1, n \rangle \in V_2$. We will assign labels to these edges. The first one will be labeled by 0 and the second one will be labeled by 1. These edge labels play no role in the game on our graph; however, it will be convenient to refer to them later.
3. From $\langle q, a, n \rangle \in V_2$ there exit two edges defined as follows: Let c be 1 if $n \in \mathbf{P}$ and 0 if $n \notin \mathbf{P}$; and for $b \in \{0,1\}$ Let q_b be $\delta_\mathcal{A}(q, \langle a, b, c \rangle)$. One edge from $\langle q, a, n \rangle$ is connected to $\langle q_0, n+1 \rangle$ and the second one to $\langle q_1, n+1 \rangle$. We label the first edge by 0 and the second one by 1.

The color of a node of the arena is defined by the color of its automaton's component, i.e., $c(\langle q, n \rangle) = c(\langle q, a, n \rangle = col(q)$.

Every node of the game graph for $G_{\mathcal{A},\mathbf{P}}$ has two successors. The subsets of V_1 (respectively, of V_2) can be identified with the memoryless strategies of Player I (respectively, of Player II). For a subset $U_1 \subseteq V_1$, the corresponding memoryless strategy f_{U_1} is defined as

$$f_{U_1}(\langle q, n \rangle) = \begin{cases} \langle q, 1, n \rangle & \text{if } \langle q, n \rangle \in U_1 \\ \langle q, 0, n \rangle & \text{otherwise} \end{cases}$$

In other words, for $v \in V_1$ the strategy f_{U_1} chooses the nodes reachable from v by the edge with label $U_1(v)$.

To a subset $U_1 \subseteq V_1$, correspond a function $h_{U_1} : \{0,1\}^* \to V_1$ and a SC-operator $F_{U_1} : \{0,1\}^\omega \to \{0,1\}^\omega$. First, we provide the definition for h_{U_1}, and later for F_{U_1}.

h_{U_1} is defined as follows:

$$h_{U_1}(\epsilon) = \langle q_{init}, 0 \rangle$$

$$\text{For } \pi \in \{0,1\}^* \text{ and } b \in \{0,1\} : h_{U_1}(\pi b) = \langle \delta_{\mathcal{A}}(q, a, b, c), n+1 \rangle, \text{ where}$$

$$\langle q, n \rangle = h_{U_1}(\pi),\ a = 1 \text{ iff } \langle q, n \rangle \in U_1 \text{ and } c = 1 \text{ iff } n \in \mathbf{P}.$$

Some explanation might be helpful at this point. Let G_{U_1} be the subgraph of $G_{\mathcal{A},\mathbf{P}}$ obtained by removing from every node $v \in V_1$ the edge labeled by $\neg U_1(v)$ and removing the label from the other edge exiting v. In this graph every V_1 node has outdegree one and every V_2 node has two exiting edges; one is labeled by 0 and the other is labeled by 1. For every π in $\{0,1\}^*$ there is a unique path from $\langle q_{init}, 0 \rangle$ to a state $v_1 \in V_1$ such that π is the sequence of labels on the edges of this path; this node v_1 is h_{U_1} image of π.

Now a SC-operator $F_{U_1} : \{0,1\}^\omega \to \{0,1\}^\omega$ which corresponds to U_1 is defined as follows. Let $\pi = b_0 b_1 \ldots$ be an ω-string. There is a unique ω-path ρ from $\langle q_{init}, 0 \rangle$ in G_{U_1} such that π is the sequence labels on the edges of this path. Let $v_1 v_2 \ldots$ be the sequence of V_1 nodes on ρ and let $a_i = 1$ if $v_i \in U_1$ and 0 otherwise. The ω sequence $a_0 a_1 \ldots$ is defined as the F_{U_1} image of π.

Similarly, $U_2 \subseteq V_2$ corresponds to a memoryless strategy f_{U_2} for Player II and C-operator F_{U_2}. The following lemmas are easy

Lemma 6. *1. Let U_1 be a subset of V_1. The memoryless strategy defined by U_1 is winning for Player I iff $R_{\mathcal{A}}(F_{U_1}(Y), Y, \mathbf{P})$ holds for every Y.*
2. Let U_2 be a subset of V_2. The memoryless strategy defined by U_2 is winning for Player II iff $\neg R_{\mathcal{A}}(X, F_{U_2}(X), \mathbf{P})$ holds for every X.

Lemma 7. *If $\mathbf{P}$ is recursive, then the game $G_{\mathcal{A},\mathbf{P}}$ is recursive. If in addition $U_1 \subseteq \mathcal{Q} \times Nat$ (respectively, $U_2 \subseteq \mathcal{Q} \times \{0,1\} \times Nat$) is recursive, then the corresponding operator F_{U_1} (respectively, F_{U_2}) is recursive.*

Recall that every subset U of $\mathcal{Q} \times Nat$ corresponds to a tuple $W_1, \ldots, W_{|\mathcal{Q}|}$ of subsets of Nat such that $\langle q, m \rangle \in U$ iff $m \in W_q$.

The next Lemma shows that the set of winning memoryless strategies for each of the players in $G_{\mathcal{A},\mathbf{P}}$ is definable in the structure $\langle Nat, <, \mathbf{P} \rangle$.

Lemma 8. *Let $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta_{\mathcal{A}}, q_{init}, col \rangle$, be a deterministic parity automaton over the alphabet $\Sigma = \{0,1\} \times \{0,1\} \times \{0,1\}$.*

1. There is an MLO formula $Win_{\mathcal{A}}(Z_1, \ldots, Z_{|\mathcal{Q}|}, Z)$ such that for every $\mathbf{P} \subseteq Nat$ and $W_1, \ldots, W_{|\mathcal{Q}|} \subseteq Nat$:
$Nat \models Win_{\mathcal{A}}(W_1, \ldots, W_{|\mathcal{Q}|}, \mathbf{P})$ iff the corresponding subset $U \subseteq \mathcal{Q} \times Nat$ defines a winning memoryless strategy for Player I in $G_{\mathcal{A},\mathbf{P}}$.

2. *There is an MLO formula* $Lose_{\mathcal{A}}(Z_1, \dots, Z_{|\mathcal{Q}|}, Z'_1, \dots, Z'_{|\mathcal{Q}|}, Z)$ *such that for every* $\mathbf{P} \subseteq Nat$ *and* $W_1, \dots, W_{|\mathcal{Q}|}, W'_1, \dots, W'_{|\mathcal{Q}|} \subseteq Nat$*:*

   $Nat \models Lose_{\mathcal{A}}(W_1, \dots, W_{|\mathcal{Q}|}, W'_1, \dots, W'_{|\mathcal{Q}|}, \mathbf{P})$ *iff the corresponding subset* $U \subseteq \mathcal{Q} \times \{0,1\} \times Nat$ *defines a winning memoryless strategy for Player II in* $G_{\mathcal{A},\mathbf{P}}$.
3. *Moreover, there is an algorithm that computes formulas* $Win_{\mathcal{A}}$ *and* $Lose_{\mathcal{A}}$ *from* $\mathcal{A}$.

Proof. (Sketch) The game arena $G_{\mathcal{A},\mathbf{P}}$ can be considered as a logical structure M for the signature $\tau_{\mathcal{A}} = \{R_i \ : \ i \in \mathcal{Q} \cup \mathcal{Q} \times \{0,1\}\} \cup \{P, E_0, E_1\}$, where R_i and P are unary predicates and $E_0,\ E_1$ are binary predicates with the interpretation

$$\mathbf{R}_i^M = \begin{cases} \{\langle q, j\rangle \ : \ j \in Nat\} & \text{for } i = q \in \mathcal{Q} \\ \{\langle q, a, j\rangle \ : \ j \in Nat\} & \text{for } i = \langle q, a\rangle \in \mathcal{Q} \times \{0,1\} \end{cases}$$

$$\mathbf{P}^M = \{\langle q, m\rangle \ : \ m \in \mathbf{P}\} \cup \{\langle q, a, m\rangle \ : \ a \in \{0,1\} \text{ and } m \in \mathbf{P}\}$$

$$\mathbf{E}_0^M(v_1, v_2) \text{ (respectively, } \mathbf{E}_1^M(v_1, v_2)) \text{ holds}$$

iff there is an edge labeled by 0 (respectively, by 1) from v_1 to v_2.

Every set S of nodes in M corresponds to the tuple $\langle \dots, W_i, \dots \rangle$ where $i \in \mathcal{Q} \cup \mathcal{Q} \times \{0,1\}$) of subsets of Nat such that $\langle q, m\rangle \in S$ iff $m \in W_q$ and $\langle q, a, m\rangle \in S$ iff $m \in W_{\langle q,a\rangle}$.

It can be shown that there is an algorithm which for every formula $\psi(X)$ in the second order monadic logic over the signature $\tau_{\mathcal{A}}$ with a free monadic variable X constructs a formula $\varphi(Y, \dots, Z_i, \dots)$ with free monadic variables $\{Y\} \cup \{Z_i \ : \ i \in \mathcal{Q} \cup \mathcal{Q} \times \{0,1\}\}$, such that for every $\mathbf{P} \subseteq Nat$ and a tuple $\langle \dots, W_i, \dots \rangle$ where $i \in \mathcal{Q} \cup \mathcal{Q} \times \{0,1\}$) of subsets of Nat the following equivalence holds:

$$Nat \models \varphi(\mathbf{P}, \dots, W_i, \dots), \text{ iff } M \models \psi(S),$$

where S is the subset of nodes in $G_{\mathcal{A},\mathbf{P}}$, which corresponds to $\langle \dots, W_i, \dots \rangle$.

Lemma 8 follows from the existence of the above algorithm and from the observation that the set of subsets of $G_{\mathcal{A},\mathbf{P}}$ which correspond to memoryless winning strategies for Player I (respectively, Player II) in $G_{\mathcal{A},\mathbf{P}}$ can be defined by a second order monadic formula $\psi(X)$. □

Finally, note that the operator $\lambda\langle U_1, X\rangle.F_{U_1}(X)$ is causal both in U_1 and in X and its construction is uniform in $\mathbf{P}$. More precisely,

Lemma 9. *1. Let* $\mathcal{A} = \langle \mathcal{Q}_{\mathcal{A}}, \Sigma, \delta_{\mathcal{A}}, q_{init}, col\rangle$ *be a finite parity automaton over the alphabet* $\Sigma = \{0,1\} \times \{0,1\} \times \{0,1\}$. *There is a finite state Mealey automaton* $\mathcal{B} = \langle \mathcal{Q}_{\mathcal{B}}, \Sigma_{\mathcal{B}}, \delta_{\mathcal{B}}, q'_{init}, \Delta, out\rangle$, *where* $\Sigma_{\mathcal{B}} = \{0,1\}^{|\mathcal{Q}_{\mathcal{A}}|} \times \{0,1\} \times \{0,1\}$ *and* $\Delta = \{0,1\}$, *such that for every* $\mathbf{P} \subseteq Nat$ *and every subset* $U_1 \subseteq \mathcal{Q}_{\mathcal{A}} \times Nat$ *of the first Player's positions in the game* $G_{\mathcal{A},\mathbf{P}}$

the SC-operator $\lambda Y.F_{U_1}(Y)$ defined by U_1 in $G_{\mathcal{A},\mathbf{P}}$ is the same as the operator $\lambda Y.F_{\mathcal{B}}(W_1, W_2, \ldots, W_{|\mathcal{Q}_\mathcal{A}|}, Y, \mathbf{P})$, where the tuple $W_1, \ldots, W_{|\mathcal{Q}_\mathcal{A}|} \subseteq Nat$ corresponds to U_1.

Moreover, $\mathcal{B}$ is computable from $\mathcal{A}$.

2. *The assertion similar to (1) for the C-operators defined by the subsets of the second Player's positions.*

4 Proof of Theorem 2

Our proof of Theorem 2 is organized as follows:

1. The implications (1)⇒(4), (2)⇒(4) and (3)⇒(4) follow from Lemma 11.
2. The implication (4)⇒(5) follows from Theorem 14.
3. The implications (5)⇒(1), (5)⇒(2) and (5)⇒(3) are proved in Lemma 15.

First note

Theorem 10. *For every $\mathbf{P} \subseteq Nat$ and every MLO formula $\psi(X, Y, Z)$ either there is a C-operator F such that $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ or there is a SC-operator G such that $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$.*

Proof. Let $\mathcal{A}$ be a parity automaton equivalent to $\neg\psi(Y, X, Z)$. By Theorem 5, one of the players has a memoryless winning strategy in the game $G_{\mathcal{A},\mathbf{P}}$. A memoryless winning strategy U of Player I (respectively, of Player II) defines SC-operator (respectively, C-operator) F_U. Hence, by Lemma 6, either there is a C-operator F such that $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ or there is a SC-operator G such that $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$.

Lemma 11. *If one of the Problems 1-5 is computable for $\mathbf{P}$, then the monadic theory of $\langle Nat, <, \mathbf{P}\rangle$ is decidable.*

Proof. Let $\beta(P)$ be a sentence in *MLO* and let $\psi_\beta(X, Y, P)$ be defined as $\big(\beta{\rightarrow}(Y = \{0\})\big) \wedge \big(\neg\beta{\rightarrow}(X = \emptyset)\big)$.

Observe that $Nat \models \beta(\mathbf{P})$ iff there is a C-operator F such that $Nat \models \forall X \psi_\beta(X, F(X, \mathbf{P}), \mathbf{P})$ iff $Nat \models \forall X \psi_\beta(X, H(X, \mathbf{P}), \mathbf{P})$ where H is a constant C-operator defined as $H = \lambda\langle X, P\rangle.10^\omega$

Hence, if one of the Problems 1-5 is computable for $\mathbf{P}$, then we can decide whether $Nat \models \beta(\mathbf{P})$. □

The proof of Lemma 11 also implies that if the following Problem 1′ is decidable for $\mathbf{P}$, then the monadic theory of $\langle Nat, <, \mathbf{P}\rangle$ is decidable.

Decision Problem 1′ for $\mathbf{P} \subseteq Nat$
Input: an *MLO* formulas $\psi(X, Y, P)$.
Question: Check whether there is a C-operator $Y = F(X, P)$ such that $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$.

Problem 1′ is actually Problem 1 without construction part.

The implication (4)⇒(5) of Theorem 2 is its difficult part. Its proof relies on the following Lemmas:

Lemma 12. *If the monadic theory of* $M = \langle Nat, <, \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ *is decidable, then* $\mathbf{P}_1, \ldots, \mathbf{P}_n$ *are recursive.*

Lemma 13. *If* $M = \langle Nat, <, \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle$ *is recursive and* $M \models \exists X_1 \ldots \exists X_m \alpha(X_1, \ldots, X_m)$*, then there are recursive sets* $\mathbf{S}_1, \ldots, \mathbf{S}_m$ *such that* $M \models \alpha(\mathbf{S}_1, \ldots, \mathbf{S}_m)$*. Moreover, if the monadic theory of* M *is decidable, then there is an algorithm for computing (programs for)* $\mathbf{S}_1, \ldots, \mathbf{S}_m$ *from* α*.*

Lemma 12 is trivial. Lemma 13 is stated by Siefkes (cf. Lemma 3 [Sie75]) without "Moreover clause". It was shown in [Sie75] that there is no algorithm that computes programs for $\mathbf{S}_1, \ldots, \mathbf{S}_m$ from programs for $\mathbf{P}_1, \ldots, \mathbf{P}_n$ and α. The "Moreover clause" was proved in [Rab05]. The algorithm in [Rab05] can even cover arbitrary unary predicates $\mathbf{P}_1, \ldots, \mathbf{P}_n$ and not only the recursive ones. It is effective when given an oracle which supplies the truth values of any *MLO* sentence on M.

Now the implication (4)⇒(5) is a consequence of Lemma 12 and the following

Theorem 14. *Let* $\mathbf{P}$ *be a unary recursive predicate over Nat. For every MLO formula* $\psi(X, Y, \mathbf{P})$ *either there is a recursive C-operator* F *such that* $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ *or there is a recursive SC-operator* G *such that* $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$*. Moreover, if the monadic theory of* $\langle Nat, <, \mathbf{P} \rangle$ *is decidable, then it is decidable which of these cases holds and the (description of the) corresponding operator is computable from* ψ*.*

Proof. Let $\varphi(X, Y, Z)$ be the formula obtained from $\psi(X, Y, P)$ when the predicate name P is replaced by variable Z. Let $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta_{\mathcal{A}}, q_{init}, col \rangle$, be a deterministic parity automaton equivalent to φ. By Theorem 5, one of the players has a memoryless winning strategy in the parity game $G_{\mathcal{A},\mathbf{P}}$.

Assume that Player I has a memoryless winning strategy. Then by Lemma 8(2), there are $W_1, \ldots, W_{|\mathcal{Q}|} \subseteq Nat$ such that $Nat \models Win_{\mathcal{A}}(W_1, \ldots, W_{|\mathcal{Q}|}, \mathbf{P})$. By Lemma 13, there are recursive $W_1, \ldots, W_{|\mathcal{Q}|} \subseteq Nat$ such that $Nat \models Win_{\mathcal{A}}(W_1, \ldots, W_{|\mathcal{Q}|}, \mathbf{P})$. Moreover, programs for $W_1, \ldots, W_{|\mathcal{Q}|}$ are computable from ψ. Hence, the corresponding winning strategy U_1 for Player I in $G_{\mathcal{A},\mathbf{P}}$ is recursive. Therefore, by Lemma 7, F_{U_1} is recursive and $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$ holds by Lemma 6 and the definition of $\mathcal{A}$.

Similar arguments show that in the case when Player II has a memoryless winning strategy, there is a recursive C-operator F_{U_2} such that $Nat \models \forall X \psi(X, F(X), \mathbf{P})$. □

Finally, we have

Lemma 15. *The implications (5)⇒(1), (5)⇒(2) and (5)⇒(3) hold.*

Proof. Let $\psi(X, Y, P)$ be a formula. By (5) either there is a recursive C-operator F such that $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ or there is a recursive SC-operator G such that $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$. Moreover, it is decidable which of these cases holds and the corresponding operator is computable from ψ.

In the first case, the answer to Problems 1-3 is positive and F is a corresponding operator.

In the second case, the answer to Problems 1-3 is negative.

Indeed, for the sake of contradiction, assume that there is a C-operator (even non-recursive) F such that $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$. Observe that F is a C-operator and G is a SC-operator. Hence, $H = \lambda X.G(F(X, \mathbf{P})$ is a SC-operator. Every SC-operator has a fixed point. Let $\mathbf{X}_0$ be a fixed point of H and let $\mathbf{Y}_0 = F(\mathbf{X}_0, \mathbf{P})$. Then we have: $\mathbf{X}_0 = G(\mathbf{Y}_0)$. Therefore, we obtain

$$Nat \models \psi(\mathbf{X}_0, \mathbf{Y}_0, \mathbf{P})$$

because $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$, and $Nat \models \neg\psi(\mathbf{X}_0, \mathbf{Y}_0, \mathbf{P})$ because $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$. Contradiction. □

5 Finite State Synthesis Problems with Parameters

Recall that a predicate $\mathbf{P} \subseteq Nat$ is ultimately periodic if there is $p, d \in Nat$ such that $(n \in \mathbf{P} \leftrightarrow n+p \in \mathbf{P})$ for all $n > d$. Ultimately periodic predicates are *MLO* definable. Hence, for every ultimately periodic predicate $\mathbf{P}$ the monadic theory of $\langle Nat, <, \mathbf{P}\rangle$ is decidable.

The next theorem implies Theorem 3 and shows that Theorem 1 can be extended only to ultimately periodic predicates.

Theorem 16. *Let* $\mathbf{P}$ *be a subset of Nat. The following conditions are equivalent and imply computability of Problem 4:*

1. $\mathbf{P}$ *is ultimately periodic.*
2. *For every MLO formula* $\psi(X, Y, P)$ *either there is a finite state C-operator* F *such that* $Nat \models \forall X \psi(X, F(X, \mathbf{P}), \mathbf{P})$ *or there is a finite state C-operator* G *such that* $Nat \models \forall Y \neg\psi(G(Y, \mathbf{P}), Y, \mathbf{P})$.
3. $\mathbf{P}$ *satisfies the following selection condition:*
   For every formula $\alpha(X, P)$ *such that* $Nat \models \exists X \alpha(X, \mathbf{P})$ *there is a finite state C-operator* $H : \{0,1\}^\omega \to \{0,1\}^\omega$ *such that* $Nat \models \alpha(H(\mathbf{P}), \mathbf{P})$.

Proof. The implication (1)⇒ (2) follows from Theorem 1 and the fact that every ultimately periodic predicate is definable by an *MLO* formula. The implication (2)⇒(3) is trivial.

The implication (3)⇒(1) is derived as follows. Let $\alpha(X, P)$ be $\forall t\big(X(t) \leftrightarrow P(t+1)\big)$. Note $Nat \models \exists X \alpha(X, \mathbf{P})$ for every $\mathbf{P} \subseteq Nat$. Therefore, if $\mathbf{P}$ satisfies selection condition, then there is C-operator $H : \{0,1\}^\omega \to \{0,1\}^\omega$ such that $Nat \models \alpha(H(\mathbf{P}), \mathbf{P})$.

Assume that a finite state Moore automaton $\mathcal{A}$ computes H and has n states. We are going to show that $\mathbf{P}$ is ultimately periodic with period at most $2n+1$. For $i \in Nat$ let a_i be one if $i \in \mathbf{P}$ and a_i be zero otherwise. Let $q_0 q_1 \dots q_{2n+1} \dots$ be the sequence states passed by $\mathcal{A}$ on the input $a_0 a_1 \dots a_{2n+1} \dots$. There are $i < j < 2n$ such that $a_i = a_j$ and $q_i = q_j$. Observe that $q_{i+1} = \delta_{\mathcal{A}}(q_i, a_i) = \delta_{\mathcal{A}}(q_j, a_j) = q_{j+1}$ and $a_{i+1} = out_{\mathcal{A}}(q_i, a_i) = out_{\mathcal{A}}(q_j, a_j) = a_{j+1}$. And by induction we get that $q_{i+m} = q_{j+m}$ and $a_{i+m} = a_{j+m}$ for all $m \in Nat$. Therefore, $\mathbf{P}$ is an ultimately periodic with a period $j - i < 2n$.

Note that this theorem does not imply that Problem 4 is computable only for ultimately periodic predicates. The next theorem can be established by the same arguments.

Theorem 17. *The following conditions are equivalent and imply computability of Problem 5:*

1. $\mathbf{P}$ *is ultimately periodic.*
2. *For every MLO formula* $\psi(X, Y, P)$ *either there is a finite state C-operator* F *such that* $Nat \models \forall X \psi(X, F(X), \mathbf{P})$ *or there is a finite state SC-operator* G *such that* $Nat \models \forall Y \neg\psi(G(Y), Y, \mathbf{P})$*. Moreover, it is decidable which of these cases holds and the corresponding operator is computable from* ψ*.*

6 Conclusion and Related Work

We investigated the Church synthesis problem with parameters. We provided the necessary and sufficient conditions for computability of Synthesis problems 1-3.

Rabin [Rab72] provided an alternative proof for computability of the Church synthesis problem. This proof used an automata on infinite trees as a natural tool for treating the synthesis problem. A C-operator $F : \{0,1\}^\omega \to \{0,1\}^\omega$ can be represented by a labelled infinite full binary tree $\langle T_2, <, \mathbf{S}\rangle$, where $\mathbf{S}$ is a subset of the tree nodes. Namely, the branches of the tree represent $X \in \{0,1\}^\omega$ and the sequence of values assigned by $\mathbf{S}$ to the nodes along the branch X represents $F(X) = Y \in \{0,1\}^\omega$. Also, the fact that $\mathbf{S}$ represents a C-operator F which uniformizes $\varphi(X, Y)$ can be expressed by an *MLO* formula $\psi(Z)$ (computable from $\varphi(X, Y)$): $T_2 \models \psi(\mathbf{S})$ iff $Nat \models \forall \varphi(X, F_{\mathbf{S}}(X))$, where $F_{\mathbf{S}}$ is the C-operator that corresponds to $\mathbf{S}$. Hence, the question whether there exists a C-operator which uniformizes φ is reduced to the problem whether $T_2 \models \exists Z \psi(Z)$. Now, the Rabin basis theorem states that if $T_2 \models \exists Z \psi(Z)$ then there is a regular subset $\mathbf{S} \subseteq T_2$ such that $T_2 \models \psi(\mathbf{S})$. The C-operator which corresponds to a regular set $\mathbf{S}$ is computable by a finite state automaton. Hence, the Büchi and Landweber theorem is obtained as a consequence of the decidability of the monadic logic of order of the full binary tree and the basis theorem.

One could try to apply the Rabin method to the Church synthesis problem with parameters. The reduction which is similar to Rabin's reduction shows that the decidability of Problem $1'$ for $\mathbf{P} \subseteq Nat$ is reduced to the decidability of the monadic theory of the labelled full binary tree $\langle T_2, <, \mathbf{Q}\rangle$ where a node is in $\mathbf{Q}$, if its distance from the root is in $\mathbf{P}$. The decidability of the latter problem can be reduced by Shelah-Stupp Muchnick Theorem [Shel75, Wal02, Th03] to the decidability of $\langle Nat, <, \mathbf{P}\rangle$. Now in order to establish computability of problems 1-3, one can try to prove the basis theorem for $\langle T_2, < .\mathbf{Q}\rangle$. Unfortunately, arguments similar to the proof of Theorem 16 show that for $\mathbf{P}, \mathbf{Q}$, as above, the following version of the basis theorem

> for every $\psi(Z, Q)$ such that $T_2 \models \exists Z \psi(Z, \mathbf{Q})$ there is a finite state operator $F(Y, U)$ such that the set which corresponds to the C-operator $\lambda X F(X, \mathbf{P})$ satisfies $\psi(Z, \mathbf{Q})$

holds only for ultimately periodic $\mathbf{P}$. Our proof of Theorem 2 implies that if the monadic theory of $\langle Nat, <, \mathbf{P}\rangle$ is decidable, then "finite state" can be replaced by "recursive" in the above version of the basis theorem.

Open Question: Are the following assertions equivalent?

1. The monadic theory of $\langle T_2, < \mathbf{S}\rangle$ is decidable.
2. For every $\psi(Z, U)$ such that $T_2 \models \exists Z \psi(Z, \mathbf{S})$ there is a recursive set $\mathbf{Q} \subset T_2$ such that $T_2 \models \psi(\mathbf{Q}, \mathbf{S})$.

Siefkes [Sie75] proved that there is a recursive set $\mathbf{S}$ for which the second assertion fails.

The conditions of Theorem 16 and Theorem 17 are sufficient, but are not necessary for computability of Synthesis problems 4-5. For example, let $\mathbf{Fac} = \{n! \ : \ n \in Nat\}$ be the set of factorial numbers. We can show that Problems 4-5 are computable for this predicate $\mathbf{Fac}$ [Rab06]. These computability results can be extended to the class of predicates for which decidability of the monadic theory of $\langle Nat, <, \mathbf{P}\rangle$ was shown by Elgot and Rabin [ER66]. Among these predicates are the sets of k-th powers $\{n^k \ : \ n \in Nat\}$ and the sets $\{k^n \ : \ n \in Nat\}$ (for $k \in Nat$). We can also show that Problems 4-5 are computable for each unary predicate in the class $\mathcal{K}$ considered by Carton and Thomas [CT02].

It is an open question whether decidability of $\langle Nat, <, \mathbf{P}\rangle$ is a sufficient condition for computability of Synthesis problems 4-5.

Games over pushdown graphs and operators computable by pushdown automata were recently studied in the literature [Th95, Wal01, CDT02, GTW02, Se04]. It is a natural question to consider the synthesis Problems 1-3 where "recursive" is replaced by "computable by pushdown automata".

Kupferman and Vardi [KV97] considered the synthesis problem with incomplete information for the specifications described by temporal logics LTL and CTL*. Their main results deal with the complexity of this synthesis problem. The decidability of the synthesis problem with incomplete information for LTL (respectively, for CTL*) can be easily derived from the Büchi-Landweber (respectively, Rabin) theorem. It seems that there are no interesting connections between the synthesis problems with incomplete information and the synthesis problems with parameters considered here.

In [RT98] a program for the relativization of finite automata theory was proposed. Our results can be seen as the first step in this direction. This step corresponds to the case where oracles are C-operators without inputs.

Acknowledgments. I would like to thank Wolfgang Thomas for bringing to my attention the paper by D. Siefkes [Sie75]. I am grateful to the anonymous referees for their suggestions.

References

[Bu60] J. R. Büchi. On a decision method in restricted second order arithmetic In *Proc. International Congress on Logic, Methodology and Philosophy of Science*, E. Nagel at al. eds, Stanford University Press, pp 1-11, 1960.

[BL69] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finitestate strategies. Transactions of the AMS, 138(27):295–311, 1969.

[CDT02] T. Cachat, J. Duparc, and W. Thomas. Solving pushdown games with a Σ_3 winning condition. In CSL02, LNCS vol. 2471, pp. 322-336, 2002.

[CT02] O. Carton and W.Thomas. The Monadic Theory of Morphic Infinite Words and Generalizations. Inf. Comput. 176(1), pp. 51-65, 2002.

[Ch69] Y. Choueka. Finite Automata on Infinite Structure. Ph.D Thesis, Hebrew University, 1970.

[ER66] C. Elgot and M. O. Rabin. Decidability and Undecidability of Extensions of Second (First) Order Theory of (Generalized) Successor. J. Symb. Log., 31(2), pp. 169-181, 1966.

[EJ91] E. A. Emerson, C. S. Jutla: Tree Automata, Mu-Calculus and Determinacy (Extended Abstract) FOCS91: 368-377, 1991.

[GTW02] E. Grädel, W. Thomas and T. Wilke. Automata, Logics, and Infinite Games, LNCS 2500, 2002.

[KV97] O. Kupferman and M.Y. Vardi, Synthesis with incomplete information, In 2nd International Conference on Temporal Logic, pp 91–106, 1997.

[PP04] D. Perrin and J. E. Pin. Infinite Words Automata, Semigroups, Logic and Games. Pure and Applied Mathematics Vol 141 Elsevier, 2004.

[Rab72] M. O. Rabin. Automata on Infinite Objects and Church's Problem Amer. Math. Soc. Providence, RI, 1972.

[RT98] A. Rabinovich and B.A. Trakhtenbrot From Finite Automata toward Hybrid Systems Proceddings of Fundamentals of Computation Theory. Lecture Notes in Computer Science 1450, pp. 411-422, Springer, 1998.

[Rab05] A. Rabinovich. On decidability of Monadic logic of order over the naturals extended by monadic predicates. Submitted, 2005.

[Rab06] A. Rabinovich. The Church problem over ω expanded by factorial numbers. In preparation, 2006.

[Sem84] A. Semenov. Logical theories of one-place functions on the set of natural numbers. Mathematics of the USSR - Izvestia, vol. 22, pp 587-618, 1984.

[Rob58] R. M. Robinson. Restricted Set-Theoretical Definitions in Arithmetic. In Proceedings of the AMS Vol. 9, No. 2. pp. 238-242, 1958.

[Se04] O. Serre. Games With Winning Conditions of High Borel Complexity. In ICALP 2004, LNCS volume 3142, pp. 1150-1162, 2004.

[Shel75] S. Shelah. The monadic theory of order. *Ann. of Math.* **102**:379–419, 1975.

[Sie75] D. Siefkes. The recursive sets in certain monadic second order fragments of arithmetic. Arch. Math. Logik, pp71-80, 17(1975).

[Th75] W. Thomas. Das Entscheidungsproblem für einige Erweiterungen der Nachfalger-Arithmetic. Ph. D. Thesis Albert-Ludwigs Universität, 1975.

[Th95] W. Thomas. On the synthesis of strategies in infinite games. In STACS '95, LNCS vo. 900, pp. 1-13. 1995.

[Th03] W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In MFCS03, LNCS 2747, 2003.

[Trak61] B. A. Trakhtenbrot. Finite automata and the logic of one-place predicates. (Russian version 1961). In AMS Transl. 59, 1966, pp. 23-55.

[Wal01] I. Walukiewicz. Pushdown processes: games and model checking. Information and Computation 164 pp. 234-263, 2001.

[Wal02] I. Walukiewicz. Monadic second order logic on tree-like structures.TCS 1:275, pp 311-346, 2002.