# Finite-Delay Strategies In Infinite Games

von
Wenyun Quan
Matrikelnummer: 253891

Diplomarbeit
im Studiengang Informatik

Betreuer: Prof. Dr. Dr.h.c. Wolfgang Thomas

Lehrstuhl für Informatik 7
Logik und Theorie diskreter Systeme
Professor Dr. Dr.h.c. Wolfgang Thomas

Fakultät für Mathematik, Informatik und Naturwissenschaften
Rheinisch-Westfälische Technische Hochschule Aachen

## Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Aachen, den 26. Juni 2009 Wenyun Quan

**Abstract**

In this thesis we study Church's Problem in the framework of infinite games between player $I$ (input) and player $O$ (output), and present algorithms for solving the problem of whether finite-delay solutions for player $O$ in infinite games exist. Even and Meyer provided an algorithm to determine solutions with finite delay for sequential Boolean equations with safety winning condition. In our work we introduce special game graphs, in which winning strategies with a fixed delay and corresponding winning regions for both players can be constructed. Using this approach, solutions with a fixed delay for player $O$ in infinite games are built up. Furthermore we present algorithms for solving the problem of whether player $O$ has winning strategies with bounded delay in arbitrary regular infinite games. As a consequence, we obtain a new proof for the result of Hosch and Landweber that the existence of the solution of Church's Problem by a strategy with a bounded delay is decidable.

# Contents

# Chapter 1

# Introduction

## 1.1 History

A central task in computer science is synthesis, which builds from a specification a program which realizes it. An instance of the synthesis problem is the description of a desired condition for an input sequence from the environment and an output sequence from the computer. Given such a condition we construct a program, which generates an output sequence to guarantee that the condition is fulfilled.

**Church's Problem.** In 1957 a master problem was raised by Church regarding the infinite behavior of automata and the use of automata as transducers. Church's Problem concerns the synthesis of automata that realizes functions over infinite words [5] :

> Given a requirement which a circuit is to satisfy, we may suppose the requirement expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The synthesis problem is to find recursion equivalences representing a circuit that satisfies the given requirement, or to determine that there is no such circuit alternatively.

Church's Problem asks for a finite-state automaton which realizes the transformation of an infinite sequence $\alpha$ into another infinite sequence $\beta$, such that a requirement on $(\alpha, \beta)$, which is expressed in certain logic formula, is satisfied. Therefore to solve Church's Problem, we determine whether the

transformations of infinite words specified in a system of arithmetic are computable by finite automata. There are classes of specifications for which algorithmic synthesis is possible. The proof of the expressive equivalence between finite automata and weak monadic second-order arithmetic over the natural number ordering, was established by Büchi and Elgot [1], and also by Trakhtenbrot [19] in 1958. This result is considered as a breakthrough on the relation between automata and logic. A few years later Büchi [4] extended the Büchi-Elgot-Trakhtenbrot Theorem to the expressive equivalence between the full monadic second-order theory of the natural number ordering and finite automata over infinite sequences. Based on a result on the determinization of Büchi automata from McNaughton [11], Church's problem was solved in 1967 by Büchi and Landweber [3] for specifications in monadic second-order logic (MSO logic) over $(\mathbb{N}, <)$.

## 1.2 Background

To study Church's Problem we work with transducers in the format of deterministic Mealy automata, which is introduced in [12]. In the theory of computation, a Mealy machine is a finite state transducer that generates an output bit based on its current state and input bit.

**Mealy automata.** A Mealy automaton has the format $\mathcal{A}_{\mathcal{M}} = (S, \Sigma, \Gamma, s_0, \delta, \tau)$ where $S$ is the finite set of states, $\Sigma$ and $\Gamma$ are the input alphabet and output alphabet, respectively, $s_0$ is the initial state, $\delta : S \times \Sigma \to S$ is the transition function and $\tau : S \times \Sigma \to \Gamma$ is the output function. A transition from $p \in S$ to $\delta(p, a) \in S$ where $a \in \Sigma$ is labeled by $a/\tau(p, a)$ where $\tau(p, a) \in \Gamma$. We present the way to specify languages and the task of synthesis in the system of monadic second-order logic (MSO) over the successor structure $(\mathbb{N}, +1, <)$. It is also called sequential calculus or S1S (second-order theory of one successor). Church's Problem can be expressed as follows [17]:

> Given a specification $\varphi$ stated in sequential calculus, decide whether a Mealy automaton exists that transforms each input sequence $\alpha \in \{0,1\}^\omega$ into an output sequence $\beta \in \{0,1\}^\omega$ such that $(\mathbb{N}, +1, <) \models \varphi(\alpha, \beta)$, and if this is the case, construct such an automaton.

**Example 1.2.1.** Let $\alpha_t$ be the $t+1$st bit of an infinite input sequence $\alpha \in \{0,1\}^\omega$, and let $\beta_t$ be the $t+1$st bit of an infinite output sequence

$\beta \in \{0,1\}^\omega$ for some $t \geq 0$. We show a finite automaton from [17] which transforms $\alpha$ into $\beta$, such that the following conditions are satisfied:

1. $\forall t \ (\alpha_t = 1 \ \rightarrow \ \beta_t = 1)$;
2. $\neg\exists t \ \beta_t = \beta_{t+1} = 0$;
3. $\exists^\omega t \ \alpha_t = 0 \ \rightarrow \ \exists^\omega t \ \beta_t = 0$.

To fulfill these three conditions we require the following procedure:

1. For input bit 1 produce output bit 1;
2. For input 0 produce:
   - output bit 1 if last output bit was 0,
   - output bit 0 if last output bit was 1.

This procedure can be executed by a Mealy automaton $\mathcal{A}_\mathcal{M}$ as follows:

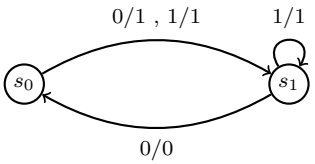


Figure 1.1: A Mealy automaton $\mathcal{A}_\mathcal{M}$

If input bit is 1, the Mealy automaton $\mathcal{A}_\mathcal{M}$ produces output bit 1 at each moment $t$. If 0 occurs in input sequence, output bit is chosen between 0 and 1 alternately.

**Remark 1.2.2.** Trakhtenbrot clarified in [18] two relevant aspects of Church's Problem. The first aspect is the computability of the transformation by a finite-state automaton, such as the Mealy automaton mentioned above. The other aspect concerns whether the transformation is nonanticipatory (i.e., causal); in other words, whether an output $\beta_t$ is determined by the prefix $\alpha_0\alpha_1 \ldots \alpha_t$ of an input $\alpha$.

In this thesis we are interested in anticipatory transformations from input sequence $\alpha$ into output sequence $\beta$. Therefore we give some examples of anticipatory transformations to understand this aspect of Church's Problem.

**Example 1.2.3.** For $\alpha, \beta \in \{0,1\}^\omega$, an anticipatory transformation can be

$$\beta_t = \alpha_{3t},$$

i.e., the output sequence $\beta$ consists of every third letter of the input sequence $\alpha$. This transformation requires response with an unbounded delay to fulfill the condition of the input sequence $\alpha$ and the output sequence $\beta$.

An instance of anticipatory transformations with a bounded delay can be

$$\beta_t = \alpha_{t+2},$$

whose transformation requires a bounded delay 3 to generate the output sequence to fulfill the certain condition.

In this thesis we do not concern anticipatory transformations with unbounded delay, but anticipatory transformations with bounded delay.

## 1.3 Motivation

In the context of reactive systems, we view finite automata with output as finite-state controllers. They are supposed to behave correctly for any behavior of the environment represented by input sequences. This amounts to solutions of infinite two-player games with winning conditions given by $\omega$-regular languages. Based on the work of Gale and Stewart [7], McNaughton introduced the treatment of Church's Problem in the framework of infinite games in [10]. Infinite games of descriptive set theory are put therefore into an algorithmic framework, and the algorithmic theory of infinite games is in a very active research area.

In our work we deal with Church's Problem in the framework of infinite games. An infinite game describes the interaction between a controller and the environment, together with a specification as a winning condition. It is more convenient to understand, therefore we consider it advantageous to transform automaton graphs into game graphs when modeling reactive systems. A specification $\varphi$ as a winning condition for player $O$ defines an infinite two-player game between player $I$ ("$I$" denotes "Input") and player $O$ ("$O$" denotes "Output"). The two players contribute respectively the input- and output-bits in turn.

In applications there are situations in which a perfect synchronized response by a controller is not possible. Hence, it is necessary to consider

possibilities with some tolerance bound, i.e., bounded delay. We search for solutions with bounded delay which fulfills conditions between two infinite sequences (i.e., binary relation). In the framework of infinite games, we ask whether or not player $O$ wins an infinite game over the corresponding finite game graph with a bounded delay, i.e., whether player $O$ wins an infinite game with delay $d$ for some $0 < d < +\infty$. The result of the work from Even and Meyer [6] provided an algorithm of solving sequential Boolean equations with a finite delay over Even-Meyer automaton graphs. In other words they solve the problem of determining finite-delay solutions in safety games. Based on this result, Hosch and Landweber [8] proved that the existence of the solution of Church's Problem with delay is decidable. We deal with Church's Problem in framework of infinite games with finite delay and provide algorithms over game graphs to find solutions with fixed delay in corresponding infinite games. The essential problem is how to determine the existence of finite-delay solutions for player $O$ in arbitrary infinite games. Our goal is to find the break condition for searching a finite-delay solution for player $O$ in regular infinite games, therefore to prove that that the existence of the solution of Church's Problem by a strategy with finite delay is decidable.

## 1.4 Outline

We start with a preliminary chapter in which we recall basic definitions relevant to our work. The idea of winning strategies with delay in infinite games will be presented as a bridge to the following chapters. The algorithm of Even and Meyer [6] is presented in Chapter 3. This algorithm provide a possibility to determine finite-delay solutions for sequential Boolean equations with safety winning condition. Furthermore we study how to construct winning strategies with a fixed delay in arbitrary infinite games over to corresponding $d$-delay game graphs in Chapter 4. The major question is how to determine the existence of solutions with finite delay in infinite game, i.e., it asks whether one of the players has a winning strategy with finite delay in an infinite game. The algorithms for solving this problem are presented in Chapter 5. Hosch and Landweber [8] proved that the problem whether Church's Problem with conditions stated in sequential calculus are solvable with delay is decidable. The conclusion of our work is that the existence of the solution of Church's Problem by a strategy with a bounded delay is decidable.

# Chapter 2

# Preliminaries

In this chapter we introduce basic definitions which are referred to in the following chapters. The terms of regular languages, infinite games and operators will be recalled, which are followed by the definition of solutions with finite delay.

## 2.1 Languages

Computers operate on texts consisting of sequences of symbols over a given alphabet. Every program transforms input texts into output texts. Therefore the basic terms are alphabets, words and languages which specify the fundamental notions of computer science. Hence, we recall in this section the definitions of regular language and $\omega$-regular language.

**Definition 2.1.1.** An *alphabet* is a finite set consisting of elements called symbols, it is denoted by $\Sigma$. Let $\Sigma^*$ be the set of all finite words over $\Sigma$ and $\Sigma^\omega$ be the set of infinite words over $\Sigma$. *A word with length n* over an alphabet $\Sigma$ is a finite sequence of letters $a_0a_1a_2\ldots a_{n-1} \in \Sigma^*$ with $a_i \in \Sigma$ for $0 \leq i \leq n-1$; an *$\omega$-word* is an infinite sequence of letters $\alpha_0\alpha_1\alpha_2\ldots \in \Sigma^\omega$ with $a_i \in \Sigma$ for $i \geq o$.

Length of a finite word $w$ is denoted by $|w|$ where $w \in \Sigma^*$. In this thesis we refer to regular languages defined by standard regular expression [9] with the operators $+, \cdot, ^*$.

**Definition 2.1.2.** An *$\omega$-regular language* is a finite union of $\omega$-languages $U \cdot V^\omega$ with regular languages $U, V \subseteq \Sigma^*$.

Let $\Sigma_I = \{0,1\}$ be an input alphabet and $\Sigma_O = \{0,1\}$ be an output alphabet. Church's Problem concerns transformations of an infinite word $\alpha = \alpha_0\alpha_1 \ldots \in \{0,1\}^\omega$ into an infinite word $\beta = \beta_0\beta_1 \ldots \in \{0,1\}^\omega$, fulfilling a certain condition $C(X,Y)$ where $X$ and $Y$ are variables for infinite sequences, i.e., $C(\alpha,\beta)$ holds. Since a condition $C(X,Y)$ can be described by an $\omega$-language $L$, Church's Problem asks whether $\alpha \times \beta := \alpha_0\beta_0\alpha_1\beta_1 \ldots \in L$ where $L \subseteq \{0,1\}^\omega$. In our work we define conditions $C(X,Y)$ in the framework of sequential calculus.

**Definition 2.1.3.** *Sequential calculus* is the class of conditions which can be expressed in a formalism which consists of:

- first order variables $x, y, z, \ldots$ interpreted as elements of the set of natural numbers $\mathbb{N}$;
- monadic second order predicate variables $X, Y, Z, \ldots$ interpreted as subsets of natural numbers $\mathbb{N}$;
- the usual logical connectives $\wedge, \vee, \neg, \ldots$;
- equality $=$;
- quantifiers over both first and second order variables $\forall, \exists$;
- the function symbol $'$ for the successor;
- the constant symbol 0 for the natural number zero.

Sequential calculus is the monadic second order theory of the natural numbers with the function successor (i.e., S1S). Church's Problem is solved by Büchi and Landweber [3] with an algorithm, which decides whether conditions stated in the framework of sequential calculus admit finite automata solutions, and produces such finite automata, if they exist.

## 2.2 Infinite games

Compared with automata graphs, game graphs offer a more intuitive view. Therefore it is useful to study Church's Problem in the framework of infinite games. Gale and Stewart [7] introduce a form of infinite games in 1953, which represents Church's Problem for any $\omega$-language $L \subseteq \{0,1\}^\omega$.

**Definition 2.2.1.** In a *Gale-Stewart game*, player $I$ and player $O$ choose symbols from an alphabet $\Sigma = \{0, 1\}$ alternately and create an infinite sequence $\alpha_0\beta_0\alpha_1\beta_1 \ldots \in \{0, 1\}^\omega$, where $\alpha_i$ denotes an input bit from player $I$ and $\beta_i$ denotes an output bit from player $O$ for $i \geq 0$.

Player $O$ wins the Gale-Stewart game if $\alpha \times \beta = \alpha_0\beta_0\alpha_1\beta_1 \ldots \in L \subseteq \{0, 1\}^\omega$, the infinite language $L$ describes the winning condition $C(X, Y)$ in the infinite game where $X$ and $Y$ are variables for infinite sequences.

Büchi and Landweber show in [3] that a condition $C(X, Y)$ stated in sequential calculus is computable by a finite automaton. An automaton can be transformed into a game graph by distinguishing the contribution of bits, which are chosen by player $I$ and player $O$. As shown in Figure 2.1 we introduce intermediate states between two steps from one step in automaton graphs, processing a single bit. Hence Gale-Stewart games with winning conditions defined in sequential calculus can be considered as two-player infinite games with finite-state game graphs.

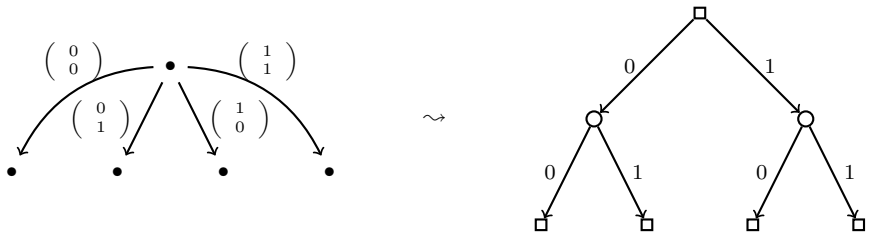


Figure 2.1: Transformation from an automaton to a game graph

We define now general game graphs with finite states for solving Church's Problem with conditions stated in sequential calculus.

**Definition 2.2.2.** A general *game graph* is a tuple $G = (Q, \mu)$ with a vertex set $Q = Q_I \dot{\cup} Q_O$ where $Q_I$ denotes the set of vertices for player $I$, $Q_O$ denotes the set of vertices for player $O$, and

$$\mu := \{Q_I \times \Sigma_I \to Q_O\} \cup \{Q_O \times \Sigma_O \to Q_I\}$$

is the function determining moves. The corresponding edge relation $E \subseteq \{Q_I \times Q_O\} \cup \{Q_O \times Q_I\}$ is defined as $(q, p) \in E$ if and only if $\mu(q, u) = p$ for some $u \in \Sigma_I \cup \Sigma_O$.

Note that all outgoing edges are connected only with vertices of the opponent. The vertices in the set $Q_I$ are depicted by boxes and the vertices in the set $Q_O$ by circles in game graphs. The set $Q$ will always be finite, therefore we also speak of finite-state game graphs.

**Definition 2.2.3.** A *play* is an infinite sequence $\rho = \rho(0)\,\rho(1)\,\rho(2)\ldots$ with $(\rho(i), \rho(i+1)) \in E$ for $i \geq 0$. The set of vertices which occur in the play $\rho$ is denoted by $\mathrm{Occ}(\rho)$:

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

$\mathrm{Inf}(\rho)$ denotes the set of vertices which occur in the play $\rho$ infinitely often:

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

If a vertex $p \in Q$ is reached by an outgoing edge from a vertex $q \in Q$ in a game graph $G$, we say that $q$ is a *predecessor* of $p$, and $p$ is a *successor* of $q$.

**Definition 2.2.4.** An *infinite game* for solving Church's Problem is a tuple $(G, q_0, \varphi)$ consisting of a game graph $G = (Q, \mu)$, an initial vertex $q_0 \in Q$ and a winning condition $\varphi$ for player $O$. The *size* of an infinite game is the number of vertices in the corresponding game graph $G$, denoted by $|Q|$.

For solving Church's Problem with conditions in sequential calculus an appropriate model of automaton is introduced by Muller [13]. The Büchi-Elgot-Trakhtenbrot Theorem shows that any MSO formula $\varphi$ can be transformed into an equivalent Muller automaton. The results in [4] and [11] allow to convert an MSO formula into an equivalent non-deterministic Büchi automaton, and hence into an equivalent deterministic Muller automaton. Thus, we recall the definition of a Muller game which is transformed from Muller automaton.

**Definition 2.2.5.** A *Muller game* is an infinite game $(G, q_0, \mathcal{F})$ with a game graph $G = (Q, \mu)$, an initial vertex $q_0 \in Q$ and a winning condition for player $O$ given by a collection $\mathcal{F} \subseteq 2^Q$. Player $O$ wins a play $\rho$ if and only if

$$\mathrm{Inf}(\rho) \in \mathcal{F}.$$

A *weak Muller game* is an infinite game $(G, q_0, \mathcal{F})$ with a game graph $G = (Q, \mu)$, an initial vertex $q_0 \in Q$ and a winning condition for player $O$ given by a collection $\mathcal{F} \subseteq 2^Q$. Player $O$ wins a play $\rho$ if

$$\mathrm{Occ}(\rho) \in \mathcal{F}.$$

**Definition 2.2.6.** A *strategy* for player $O$ in an infinite game is a function $f : (Q_I \cdot Q_O)^+ \to \Sigma_O$, i.e., it determines output bits on an infinite word $\alpha \times \beta = \alpha_0\beta_0\alpha_1\beta_1 \ldots \in \{0,1\}^\omega$ of even length; analogously a function $g : (Q_I \cdot Q_O)^* \cdot Q_I \to \Sigma_I$ is a strategy for player $I$ which determines input bits on an infinite word $\alpha \times \beta = \alpha_0\beta_0\alpha_1\beta_1 \ldots \in \{0,1\}^\omega$ of odd length. A *positional strategy* $f : Q_O \to \{0,1\}$ for player $O$ can be described by a subset of the set of edges $E$:

$$\{(q,p) \in E \mid q \in Q_O, f(q) = p \in Q_I\} \subseteq E;$$

analogously for a positional strategy $g : Q_I \to \{0,1\}$ for player $I$.

A play $\rho = \rho(0)\rho(1)\rho(2)\ldots$ started in $\rho(0)$ is *consistent with* $f$ if for every $\rho(i) \in Q_O$, $\rho(i+1) = \mu(\rho(i), f(\rho(0)\ldots\rho(i)))$ holds.

A strategy $f$ is a *winning strategy* from $q \in Q$ for player $O$ in a game graph $G = (Q, \mu)$ if each play from $q$ consistent with $f$ fulfills the winning condition $\varphi$, analogously for a winning strategy $g$ for player $I$. Let $\mathrm{Win} \subseteq Q^\omega$ be the set of the plays won by player $O$, in a game graph $G$ a *winning region* of player $O$ is defined as follows:

$$W_O := \{q \in Q \mid \text{player } O \text{ has a winning strategy from } q\}.$$

A winning region of player $I$ denoted by $W_I$ is defined analogously.

In our work we concern Church's Problem, thus we are interested in the problem of deciding whether player $O$ has a winning strategy from the initial vertex $q_0$ in a game graph $G = (Q, \mu)$, i.e., whether $q_0 \in W_O$.

**Definition 2.2.7.** A game is *determined* if each vertex of the game graph is either in $W_I$ or in $W_O$, i.e., $W_I \dot\cup W_O = Q$ holds.

Büchi and Landweber show in [3] that every infinite game with winning condition definable in sequential calculus, is determined. Hereby we recall definitions of determined infinite games with different winning conditions for player $O$.

- A *reachability game* $(G, q_0, \varphi)$ is an infinite game together with a game graph $G = (Q, \mu)$, an initial vertex $q_0 \in Q$ and a set $F \subseteq Q$ defining a reachability winning condition as follows:

  $$\rho \in \mathrm{Win} :\Leftrightarrow \exists i : \rho(i) \in F.$$

  Reachability games are a special case of weak Muller games.

- A *safety game* $(G, q_0, \varphi)$ is an infinite game together with a game graph $G = (Q, \mu)$, an initial vertex $q_0 \in Q$ and a set $F \subseteq Q$ defining a safety winning condition as follows:

$$\rho \in \text{Win} :\Leftrightarrow \forall i : \rho(i) \in F.$$

- An infinite game $(G, q_0, \varphi)$ given by a game graph $G = (Q, \mu)$, an initial vertex $q_0 \in Q$ and $F \subseteq Q$ is called a *Büchi game* if the winning condition of player $O$ is given by:

$$\rho \in \text{Win} :\Leftrightarrow \text{Inf}(\rho) \cap F \neq \emptyset.$$

- The dual co-Büchi winning condition of player $O$ is:

$$\rho \in \text{Win} :\Leftrightarrow \text{Inf}(\rho) \subseteq F,$$

  an infinite game with such a winning condition is called a *co-Büchi game*.

- Let $G = (Q, \mu)$ be a game graph and $c$ be a coloring function $c : Q \to \{0, 1, \ldots, k\}$ for some $k \in \mathbb{N}$. A play $\rho = \rho(0)\rho(1)\rho(2)\ldots$ delivers a sequence of colors $c(\rho) = c(\rho(0))\,c(\rho(1))\,c(\rho(2))\ldots$ A *weak parity game* is an infinite game $(G, q_0, \varphi)$, a game graph $G = (Q, \mu)$ with a coloring function $c : Q \to \{0, 1, \ldots, k\}$ for some $k \in \mathbb{N}$, an initial vertex $q_0 \in Q$ and a weak parity condition defining the set of plays $\rho$ won by player $O$:

$$\rho \in \text{Win} :\Leftrightarrow \max(\text{Occ}(c(\rho))) \text{ is even.}$$

- Given a game graph $G = (Q, \mu)$ with a coloring function $c : Q \to \{0, 1, \ldots, k\}$ for some $k \in \mathbb{N}$, and an initial vertex $q_0 \in Q$. A *parity game* is an infinite game $(G, q_0, \varphi)$ with a parity winning condition defining the set of plays $\rho$ won by player $O$:

$$\rho \in \text{Win} :\Leftrightarrow \max(\text{Inf}(c(\rho))) \text{ is even.}$$

  The parity winning condition is the most fundamental one of all winning conditions for infinite two-player games. Every other winning condition can be reduced to the parity condition.

## 2.3 Operators

**Definition 2.3.1.** Given two finite sets, $\Sigma_I$ as an input alphabet and $\Sigma_O$ as an output alphabet, $\Sigma_I^\omega$ and $\Sigma_O^\omega$ denote the sets of infinite words over $\Sigma_I$ and $\Sigma_O$ respectively. Let $X$ be a variable for infinite sequences in $\Sigma_I^\omega$ and $Y$ be a variable for infinite sequences in $\Sigma_O^\omega$. Let $C(X,Y)$ be a binary relation between $X$ and $Y$, and *op* be an operator mapping $\Sigma_I^\omega$ into $\Sigma_O^\omega$. We say that an operator $op : \Sigma_I^\omega \to \Sigma_O^\omega$ is a *solution for a condition* $C(X,Y)$ *for* $Y$ if

$$\{(\alpha, op(\alpha)) \mid \alpha \in \Sigma_I^\omega\} \subseteq C(X,Y).$$

An operator $op' : \Sigma_O^\omega \to \Sigma_I^\omega$ is a *solution for a condition* $\neg C(X,Y)$ *for* $X$ if

$$\{(op'(\beta), \beta) \mid \beta \in \Sigma_O^\omega\} \subseteq \neg C(X,Y).$$

**Definition 2.3.2.** An operator *op* for infinite words is called *nonanticipatory* or *causal*, if for any infinite words

$$\begin{aligned} \alpha' &= \alpha'_0\alpha'_1 \ldots \alpha'_t \ldots \text{ and} \\ \alpha'' &= \alpha''_0\alpha''_1 \ldots \alpha''_t \ldots \end{aligned}$$

for $t \geq 0$, the corresponding outputs

$$\begin{aligned} op(\alpha') &= \beta' = \beta'_0\beta'_1 \ldots \beta'_t \ldots \text{ and} \\ op(\alpha'') &= \beta'' = \beta''_0\beta''_1 \ldots \beta''_t \ldots \end{aligned}$$

fulfill the following condition:

$$\alpha'_0 = \alpha''_0, \alpha'_1 = \alpha''_1, \ldots, \alpha'_t = \alpha''_t, \ldots \Rightarrow \beta'_0 = \beta''_0, \beta'_1 = \beta''_1, \ldots, \beta'_t = \beta''_t, \ldots;$$

i.e., the output $\beta_t$ at time $t$ is uniquely determined by the input sequence $\alpha_0, \alpha_1, \ldots, \alpha_t$, and is independent of the input sequence $\alpha_{t+1}, \alpha_{t+2}, \ldots$ in the future. Otherwise an operator *op* for infinite words is an *anticipatory* operator.

Trakhtenbrot and Barzdin introduce in [15] a type of operators with delay as solutions to fulfill conditions $C(X,Y)$ between infinite words.

**Definition 2.3.3.** An operator $op : \Sigma_I^\omega \to \Sigma_O^\omega$ is *continuous* if for each $\alpha \in \Sigma_I^\omega$, each bit $\beta_t$ of $op(\alpha)$ depends only on a finite prefix of $\alpha$ for $0 \leq t < +\infty$.

Hence, every nonanticipatory operator is continuous.

**Example 2.3.4.** The following operator $op_1$ is causal:

$$op_1(\alpha_t) = \begin{cases} 1, & \text{if } \alpha_0\alpha_1 \ldots \alpha_t \text{ contains more ones than zeros;} \\ 0, & \text{otherwise.} \end{cases}$$

A continuous operator $op_2$ for some $k$ with $0 < k < +\infty$ is shown as follows:

$$op_2(\alpha_t) = \begin{cases} 1, & \text{if } \alpha_0\alpha_1 \ldots \alpha_t \ldots \alpha_{t+k} \text{ contains more ones than zeros;} \\ 0, & \text{otherwise.} \end{cases}$$

And the following operator $op_3$ is a typical non-continuous one:

$$op_3(\alpha_t) = \begin{cases} 1, & \text{if } \alpha_0\alpha_1 \ldots \alpha_t \ldots \text{ contains infinitely many ones;} \\ 0, & \text{otherwise.} \end{cases}$$

We are interested in the class of continuous operators $op$ mapping $\Sigma_I^\omega$ into $\Sigma_O^\omega$ for which

$$\beta_t = \delta\left(\alpha_0\alpha_1 \ldots \alpha_{f(t)}\right)$$

holds, where $\beta_t$ is the $(t+1)$st element of the sequence $\beta$, $f$ is a function $f : \mathbb{N} \to \mathbb{N}$, and $\delta$ maps $\Sigma_I^*$ into $\Sigma_O$ where $\Sigma_I^*$ denotes a set of all finite sequences over $\Sigma_I$.

**Definition 2.3.5.** A continuous operator $op$ is called *deterministic* if $f(t) \leq t$, i.e., if for $0 \leq d < +\infty$ it can be given in the form

$$\beta_t = \delta(\alpha_0\alpha_1 \ldots \alpha_{t+d}).$$

**Definition 2.3.6.** A deterministic operator is a *d-shift operator* for some $d \geq 0$ if $f(t) = t - d$ for $t \geq d$, otherwise $f(t) = 0$, i.e., a $d$-shift operator produces $\beta_t$ according to the sequence $\alpha_0 \ldots \alpha_{t-d}$. An operator is a *d-delay operator* for some $d > 0$ if $f(t) = t + d - 1$, which means a $d$-delay operator will not generate $\beta_t$ until a finite sequence $\alpha_0 \ldots \alpha_{t+d-1}$ is seen.

**Example 2.3.7.** We have following examples for $d$-shift operator and $d$-delay operator:

1. 3-shift operator generates

$$\beta_t = \begin{cases} 1, & \text{if } \alpha_0\alpha_1 \ldots \alpha_{t-3} \text{ contains even many ones;} \\ 0, & \text{otherwise.} \end{cases}$$

In other words player $O$ has to produce the output bit $\beta_t$ depending only on the input sequence up to time $t-3$.

| t | 0 | 1 | 2 | 3 | 4 | 5 | $\cdots$ |
|---|---|---|---|---|---|---|---|
| Player $I$ | 0 | 1 | 0 | 1 | 1 | 0 | $\cdots$ |
| Player $O$ | 0 | 0 | 0 | 1 | 0 | 0 | $\cdots$ |

2. 3-delay operator generates

$$\beta_t = \begin{cases} 1, & \text{if } \alpha_0\alpha_1 \ldots \alpha_{t+2} \text{ contains even many ones;} \\ 0, & \text{otherwise.} \end{cases}$$

In other words player $O$ can wait until the sequence of input bits from player $I$ till time $t+2$ is seen.

| t | 0 | 1 | 2 | 3 | 4 | 5 | $\cdots$ |
|---|---|---|---|---|---|---|---|
| Player $I$ | 0 | 0 | 1 | 0 | 1 | 1 | $\cdots$ |
| Player $O$ | 0 | 1 | 1 | 0 | $\cdots$ | | |

**Definition 2.3.8.** A condition $C(X,Y)$ is *determined* if and only if there exists either a 1-delay solution $\beta = op(\alpha)$ of the condition $C(X,Y)$ for $Y$ or a 1-shift solution $op'$ of the condition $\neg C(X,Y)$ for $X$.

Büchi and Landweber [3] give a synthesis algorithm for sequential calculus with respect to deterministic operators by showing that every condition $C(X,Y)$ of sequential calculus is determined.

Let $X$ be a variable for infinite sequences, we write $X(t)$ for the $t+1$st position in the corresponding sequence, analogously for infinite words $Y$ and $Z$.

**Remark 2.3.9.** A condition $C(X,Y)$ stated in sequential calculus has a $d$-delay solution for $Y$ if and only if the condition defined by the formula

$$C_d(X,Y) := \exists Z\,(C(Z,Y) \wedge \forall t\,(Z(t) \leftrightarrow Y(t+d-1)))$$

has a 1-delay solution for $Y$. It is equivalent to $C(X,Y)$ having a $d$-delay solution for $Y$. A condition $\neg C(X,Y)$ in sequential calculus has a $d$-shift solution for $X$ if and only if the condition defined by the formula

$$\neg C_d(X,Y) := \forall Z\,(\forall t\,(Z(t) = Y(t+d-1)) \rightarrow \neg C(Z,Y))$$

has a 1-shift solution for $X$. It is equivalent to $\neg C(X,Y)$ having a $d$-shift solution for $X$.

For any fixed $d$ the existence of a $d$-delay solution is determined according to the Büchi-Landweber algorithm [3]. Because conditions in sequential calculus are determined, $C_d(X,Y)$ does not have a 1-delay solution for $\beta$ if and only if $\neg C_d(X,Y)$ has a 1-shift solution for $X$. It also means that for every condition $C(X,Y)$ which is defined in sequential calculus and for every fixed $d$, either $C(X,Y)$ has a $d$-delay solution for $Y$ or $\neg C(X,Y)$ has a $d$-shift solution for $X$ [8].

**Remark 2.3.10.** If $C(X,Y)$ has a $d$-delay solution for $Y$ for some $d > 0$, it also has $d'$-delay solutions for $Y$ for all $d' > d$; and if $\neg C(X,Y)$ has a $d$-shift solution for $X$, it also has $d'$-shift solutions for all $0 \leq d' < d$.

## 2.4 Winning strategies with delay

An operator *op* for solving a condition $C(X,Y)$ can be considered as a winning strategy for player $O$ in an infinite game, which is specified by the $\omega$-language described by $C(X,Y)$. We consider a certain infinite game specified by the condition $C(X,Y)$. Player $I$ selects an element $\alpha_t$ of an input alphabet $\Sigma_I$ at time $t$, and then player $O$ makes a move by selecting an element $\beta_t$ of an output alphabet $\Sigma_O$. Both players have complete information about all previous moves of their opponent, i.e., player $I$ knows the output sequence $\beta_0\beta_1\ldots\beta_{t-1}$ when making a move at time $t$, and player $O$ replies with the input sequence $\alpha_0\alpha_1\ldots\alpha_t$. A play $(\alpha,\beta)$ of the game creates an infinite sequence $\alpha_0\beta_0\alpha_1\beta_1\ldots$ consisting of all moves of each player. Player $O$ wins the game if $C(X,Y)$ is satisfied by a play $(\alpha,\beta)$, otherwise player $I$

wins. Hence in the framework of infinite games, a strategy for player $I$ is a deterministic 1-shift operator $\eta : \Sigma_O^\omega \to \Sigma_I^\omega$ and a strategy for player $O$ is a deterministic 1-delay operator $\delta : \Sigma_I^\omega \to \Sigma_O^\omega$.

**Remark 2.4.1.** A condition $C(X, Y)$ is determined if and only if one of the players has a winning strategy in the infinite game with the winning condition $C(X, Y)$, i.e., player $I$ has a $d$-shift operator or player $O$ has a $d$-delay operator, with which they can beat all strategies of their opponent.

An operator $\delta : \Sigma_I^\omega \to \Sigma_O^\omega$ is a $d$-delay winning strategy for player $O$ in an infinite game defined by a condition $C(X, Y)$, if and only if it is a deterministic $d$-delay solution of $C(X, Y)$ for $Y$. As mentioned in the previous section, player $O$ therefore has winning strategies with all delay $d'$ for $d' > d$. Analogously for player $I$, an operator $\eta$ is a $d$-shift winning strategy for player $I$ in an infinite game described by a condition $C(X, Y)$, if and only if it is a deterministic $d$-shift solution of $C(X, Y)$ for $X$. Player $O$ has no $d$-delay winning strategy in this infinite game and player $I$ has winning strategies with all shift $d'$ for $0 \leq d' < d$.

**Example 2.4.2.** We consider the operator $op_3$ in Example 2.3.4. It shows us that player $O$ wins the game only with an infinite delay, i.e., player $O$ produces a sequence $\beta \in \Sigma_O^\omega$ according to an entire input sequence $\alpha \in \Sigma_I^\omega$ of player $I$. It is clear that no finite delay suffices for player $O$ to win. Because if player $O$ chooses output bits with a delay $d$, which means a bit $\beta_t$ is determined after the choice of $\alpha_t \alpha_{t+1} \dots \alpha_{t+d-1}$ by player $I$, player $I$ may decide the continuing sequence $\alpha_{t+d}\alpha_{t+d+1} \dots \in 0^\omega$ for $\beta_{t+d-1} = 1$, or $\alpha_{t+d}\alpha_{t+d+1} \dots \in 1^\omega$ for $\beta_{t+d-1} = 0$ respectively. In another aspect, player $I$ has a winning strategy with an arbitrary finite shift $d$, or player $I$ loses the game with an infinite shift.

In fact there are infinite games in which player $I$ wins with an infinite shift, i.e., player $O$ does not have a winning strategy even with an infinite delay in these infinite games, see the operator $op_3$ in Example 2.3.4. Our main purpose in this thesis is to determine whether a condition $C(X, Y)$ stated in sequential calculus has a solution with a finite delay for $Y$ or an infinite delay, or $\neg C(X, Y)$ has a solution with an infinite shift for $X$. In other words we look for winning strategies with delay for player $O$ in an infinite game specified by a winning condition $C(X, Y)$ for player $O$. There are three situations:

1. Player $O$ wins an infinite game with a finite delay $d$ (i.e., a bounded delay), therefore player $O$ wins also with delay $d'$ for all $d' > d$; i.e., player $I$ has winning strategies with shift $d''$ for all $d'' < d$ in the game;

2. Player $O$ has no winning strategy with delay in an infinite game, i.e., player $I$ wins an infinite game with an arbitrary shift, and also with an infinite shift;

3. Player $O$ wins an infinite game, but only with an infinite delay, i.e., player $I$ has an arbitrary finite shift winning strategy in the game.

Since we are interested only in continuous operators, we do not consider the third case. We study in the following chapters the algorithms to distinguish the first two situations in different infinite games, and focus on the problem of deciding whether a finite delay solution for player $O$ exists in an infinite game.

# Chapter 3

# Finite-Delay Solutions for Sequential Boolean Equations

In this chapter we present an algorithm of Even and Meyer which determines finite delay solutions for sequential Boolean equations.

## 3.1 Sequential Boolean equations

Even and Meyer [6] solve the finite delay problem for a fragment of sequential calculus called sequential Boolean equations. They show that the problem to decide whether such an equation has a solution with a finite delay can be reduced to the problem of deciding whether a given finite automaton graph is solvable with a finite delay. For a finite Even-Meyer automaton graph, they provide an algorithm determining whether or not a sequential Boolean equation has a finite-delay solution. Because of the equivalence between solving sequential Boolean equations and solving problem for finite automaton graphs, the problem whether a corresponding sequential Boolean equation has a solution with a finite delay is solvable.

**Definition 3.1.1.** Let $F(X, Y, dY)$ be a quantifier free formula of sequential calculus without constants, where $X$ is a vector $(x_0, x_1, \ldots, x_{m-1})$ consisting of independent Boolean variables, $Y$ is a vector $(y_0, y_1, \ldots, y_{n-1})$ consisting of dependent binary variables and $dY = (dy_0, dy_1, \ldots, dy_{n-1})$. $F$ is a Boolean expression with:

- Boolean addition '+';

- Boolean multiplication '$\cdot$';
- Complementation '$^-$';
- $dz(t) = z(t+1)$ is the value of $z$ one time unit later under the condition that time is discrete.

A *sequential Boolean equation* is written in the form

$$F(X, Y, dY) = 0.$$

**Definition 3.1.2.** The equation $F(X, Y, dY) = 0$ *has a solution* if and only if for every finite sequence of values for $X = X(0), X(1), \ldots, X(T)$ there exists a sequence of values of $Y = Y(0), Y(1), \ldots, Y(T), Y(T+1)$ such that the equation will hold for every $t = 0, 1, \ldots, T$.

**Definition 3.1.3.** The equation *has a solution with a finite delay* $d$ if the knowledge of $X(0), X(1), \ldots, X(d-1)$ is sufficient to determine a value for $Y(0)$; the knowledge of $Y(t-1), X(t-1), X(t), \ldots, X(t+d-1)$ for every $t > 0$ is sufficient to determine a value for $Y(t)$ such that the determined part of the sequence of $Y$ is part of a solution for the given sequence of $X$.

An alternative statement is that there exists operators $op_1$ and $op_2$, such that for any $T \geq d$ and any sequence $X(0), X(1), \ldots, X(T)$ the sequence $Y(0), Y(1), \ldots, Y(T-d)$ which is given by

$$\begin{aligned} Y(0) &= op_1(X(0), X(1), \ldots, X(d-1)) \\ Y(t+1) &= op_2(Y(t), X(t), X(t+1), \ldots, X(t+d)) \end{aligned}$$

for $t = 0, 1, \ldots, T-d$, can be augmented by some values for $Y(T-d+1), \ldots, Y(T)$ to satisfy the equation $F(X, Y, dY) = 0$ for every $t = 0, 1, \ldots, T$.

In the following example, we show that equations of the form $F(X, Y, dY) = 0$ are enough to represent even more general equation of the form $F'(X, dX, Y, dY) = 0$.

**Example 3.1.4.** Given the following equation with an independent variable $x$ and a dependent variable $y$:

$$x\bar{y}d\bar{y} + xydy + \bar{x}\bar{y}dx + \bar{x}yd\bar{x} = 0.$$

We can replace $dx$ with $dz$ and form the simultaneous equations as follows:

$$\begin{cases} x\bar{y}d\bar{y} + xydy + \bar{x}\bar{y}dz + \bar{x}yd\bar{z} = 0; \\ x = z. \end{cases}$$

These two simultaneous equations are equivalent, because

- $x = 0$ and $y = 0 \Leftrightarrow x + y = 0$;
- $x = y \Leftrightarrow x\bar{y} + \bar{x}y = 0$.

The resulting equation in the form $F(x, y, z, dy, dz) = 0$ is

$$x\bar{y}d\bar{y} + xydy + \bar{x}\bar{y}dz + \bar{x}yd\bar{z} + x\bar{z} + \bar{x}z = 0$$

where $x$ is an independent variable, $y$ and $z$ are both dependent variables.

## 3.2 Representing sequential Boolean equations by graphs

**Definition 3.2.1.** A *directed graph $G$ over a finite alphabet* $\Sigma$ is a system $(V, V_0, E)$ where $V$ is the finite set of vertices, $V_0 \subseteq V$ is the set of initial vertices and $E \subseteq V \times \Sigma \times V$ is the set of labeled directed edges. We can extend $E$ to edges labeled by finite words from $\Sigma^*$ as follows:

$$E(v_i, \varepsilon, v_j) \;\Leftrightarrow\; v_i = v_j$$

with $\varepsilon$ denotes empty word and

$$E(v_i, ua, v_j) \Leftrightarrow \exists v_k \left(E(v_i, u, v_k) \wedge E(v_k, a, v_j)\right)$$

with $u \in \Sigma^*$ and $a \in \Sigma$ for all $0 \leq i, j, k < |V|$. A *path* in $G$ with labels $l_0 l_1 \ldots l_{n-1} \in \Sigma^*$ is a sequence of edges of the form $(v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_{n-1}, v_n)$.

Let $G$ be a directed graph with $n$ vertices $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and each edge labeled with $l \in \Sigma$. The only restriction on $G$ is that there exists at most one edge from $v_i$ to $v_j$ with label $l \in \Sigma$ for $0 \leq i, j < n$, i.e., on each path there is a unique word consisting of labels which appear on the path.

Let $U \subseteq V$ be any set of vertices and $u \in \Sigma^*$ be a finite word, $\Delta(U, u)$ is the set of all vertices which are reachable from vertices in $U$ through all paths labeled with $u$

$$\Delta(U, u) = \{v \in V | v \in E(w, u, v) \text{ with } w \in U\}.$$

The representation of equations by graphs is best shown by means of an example.

**Example 3.2.2.** We consider the equation from Example 3.1.4

$$x\bar{y}d\bar{y} + xydy + \bar{x}\bar{y}dz + \bar{x}yd\bar{z} + x\bar{z} + \bar{x}z = 0.$$

We first show how to expand the left side on $y$, $z$, $dy$ and $dz$. For each possible value of $y$, $z$, $dy$ and $dz$ we build a term, for instance $yzdydz$ when $y, z, dy$ and $dz$ are all 1. The value of the term $yzdydz$ in brackets is determined by the above equation as follows:

$$0 + x + 0 + 0 + \bar{x} = 1,$$

thus the value in bracket for the term $yzdydz$ is 1. Furthermore we obtain the result of the expansion on the left side of this equation:

$$\begin{array}{ccccccccccc}
yzdydz\,(1) & + & yzdyd\bar{z}\,(1) & + & yzd\bar{y}dz\,(\bar{x}) & + & yzd\bar{y}d\bar{z}\,(\bar{x}) & + \\
y\bar{z}dydz\,(x) & + & y\bar{z}dyd\bar{z}\,(1) & + & y\bar{z}d\bar{y}dz\,(x) & + & y\bar{z}d\bar{y}d\bar{z}\,(1) & + \\
\bar{y}zdydz\,(\bar{x}) & + & \bar{y}zdyd\bar{z}\,(\bar{x}) & + & \bar{y}zd\bar{y}dz\,(1) & + & \bar{y}zd\bar{y}d\bar{z}\,(1) & + \\
\bar{y}\bar{z}dydz\,(1) & + & \bar{y}\bar{z}dyd\bar{z}\,(x) & + & \bar{y}\bar{z}d\bar{y}dz\,(1) & + & \bar{y}\bar{z}d\bar{y}d\bar{z}\,(x) & = & 0.
\end{array}$$

This equation can be represented as a graph. Consider the term $yzd\bar{y}dz\,(\bar{x})$ for instance, it means that $yzd\bar{y}dz = 1$ or in other words that the state $yz = 1$ may be followed by the state $d\bar{y}dz = 1$ if and only if $\bar{x} = 0$ or $x = 1$. $(y, z)$ has 4 possible values $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$, hence a graph with 4 vertices can be constructed to represent these 4 states. The vertex $(y, z)$ is connected to the vertex $(d\bar{y}, dz)$ with an edge labeled by $x$ if and

only if the left side of the equation becomes zero. For the term $yzd\bar{y}dz\,(\bar{x})$, vertex $(1,1)$ is connected to $(0,1)$ with an edge labeled by 1. The resulting graph is shown in Figure 3.1 and the table representing the graph $G$ is shown in Table 3.1.

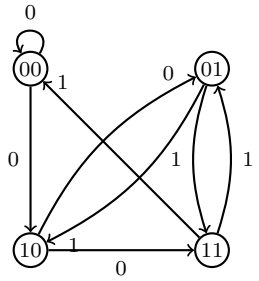


Figure 3.1: Graph $G_1$

| | 0 | 1 |
|---|---|---|
| 00 | 00, 10 | – |
| 01 | – | 10, 11 |
| 10 | 01, 11 | – |
| 11 | – | 00, 01 |

Table 3.1: Table of graph $G_1$

For any graph we can choose codes for vertices of the graph arbitrarily as long as the code of each vertex is unique. We require codes with length $k$ with $2^k \geq n$ for a graph with $n$ vertices. By $2^k > n$, the codes which is not assigned to vertices are considered as assigned to the vertices which are not connected by any edges. As those isolated vertices never appear in a sequence of values for $Y$, we can just ignore them. When $2^k = n$ we can assign each code to a vertex.

**Example 3.2.3.** Let $G_2$ in Figure 3.2 be a given graph, the graph is represented in Table 3.2 with the chosen codes for vertices of the graph $G$.

$$a:\ 00$$
$$b:\ 01$$

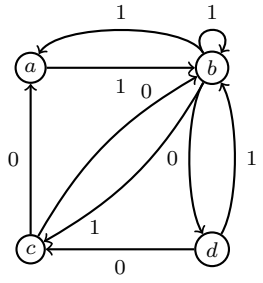


Figure 3.2: Given graph $G_2$

$$c: \ 10$$
$$d: \ 11$$

| | 0 | 1 |
|---|---|---|
| 00 | $-$ | 01 |
| 01 | 11 | $00, 01, 10$ |
| 10 | $00, 01$ | $-$ |
| 11 | 10 | 01 |

Table 3.2: Table of graph $G_2$

We can construct a corresponding equation from the table of the graph $G_2$ in Figure 3.2 which is equivalent to the equation in the form $F(X, Y, y) = 0$.

$$\begin{array}{ccccccccccc}
yzdydz\,(1) & + & yzdyd\bar{z}\,(x) & + & yzd\bar{y}dz\,(\bar{x}) & + & yzd\bar{y}d\bar{z}\,(1) & + \\
y\bar{z}dydz\,(1) & + & y\bar{z}dyd\bar{z}\,(1) & + & y\bar{z}d\bar{y}dz\,(x) & + & y\bar{z}d\bar{y}d\bar{z}\,(x) & + \\
\bar{y}zdydz\,(x) & + & \bar{y}zdyd\bar{z}\,(\bar{x}) & + & \bar{y}zd\bar{y}dz\,(\bar{x}) & + & \bar{y}zd\bar{y}d\bar{z}\,(\bar{x}) & + \\
\bar{y}\bar{z}dydz\,(1) & + & \bar{y}\bar{z}dyd\bar{z}\,(1) & + & \bar{y}\bar{z}d\bar{y}dz\,(\bar{x}) & + & \bar{y}\bar{z}d\bar{y}d\bar{z}\,(1) & = & 0.
\end{array}$$

A directed graph representing an equation in the form $F(X, Y, dY) = 0$ has $2^n$ vertices and $2^m$ edges, where $X = \langle X_0, X_1, \ldots, X_{m-1} \rangle$ and $Y =$

$\langle Y_0, Y_1, \ldots, Y_{n-1} \rangle$ are second order variables. With a sequence of $X$, namely $X(0), X(1), \ldots, X(T-1)$, a sequence of $Y$, namely $Y(0), Y(1), \ldots, Y(T)$ is determined such that the equation holds for $t = 0, 1, \ldots, T-1$. We want to find a path of length $T$ in the graph with the same labeled word as a given one. In such a path, a sequence of vertices will specify a sequence of values of $Y$ which satisfies the requirement and vice versa. Thus we say that the problem of deciding whether an equation has a solution is equivalent to the problem of deciding whether the corresponding graph is solvable. With a graph $G$ and label-alphabet $\Sigma$ we can construct a corresponding equation which has a solution if and only if the graph is solvable.

## 3.3 Solvable graphs of sequential Boolean equations

The problem of deciding whether a given graph is solvable is equivalent to the problem of deciding whether the corresponding finite-state automaton accepts all words.

**Definition 3.3.1.** *An automaton $\mathcal{A}$ corresponding to a graph $G$ with labels from $\Sigma$* is defined as follows:

1. The set of states is $V$;
2. The input alphabet is $\Sigma$;
3. The set of next states from the state $v_i$ with an input letter $l_i$ is $\delta(\{v_i\}, l_i)$;
4. The set of initial states is $V$;
5. The set of final states is $V$.

If a word is accepted by an automaton $\mathcal{A}$, then there exists a path labeled with this word in $G$. We show how to find a path in $G$ with a given word $l_0 l_1 \ldots l_{T-1} \in Sigma^*$:

1. Let $V_0$ be $V$ the set of all vertices of $G$;
2. Let $V_t = \Delta(V_{t-1}, l_{t-1})$ for every $t = 1, 2, \ldots, T-1$;

3. The result is a sequence $V_0, V_1, \ldots, V_{T-1}, V_T$ such that all vertices in $V_t$ are reachable by a path labeled with $l_0 l_1 \ldots l_{t-1}$;

4. Choose any $v_t$ from $V_t$ and find a $v_{t-1} \in V_{t-1}$ such that there is an edge labeled with $l_{t-1}$ from $v_{t-1}$ to $v_t$ for every $t = 1, 2, \ldots, T-1$;

5. The sequence $v_0 v_1 \ldots v_{t-1} v_t$ is one of the solution.

The following example illustrates the procedure of the above-mentioned construction.

**Example 3.3.2.** Given a directed graph $G_2$ in Figure 3.2, the transition table of this graph is shown in Table 3.3.

| | 0 | 1 |
|---|---|---|
| $a$ | $-$ | $b$ |
| $b$ | $d$ | $a, b, c$ |
| $c$ | $a, b$ | $-$ |
| $d$ | $c$ | $b$ |

Table 3.3: Table $T_2$ of $G_2$

The paths on $A$ labeled with the word 1010 are:

$$\{a,b,c,d\} \xrightarrow{1} \{a,b,c\} \xrightarrow{0} \{a,b,d\} \xrightarrow{1} \{a,b,c\} \xrightarrow{0} \{a,b,d\}.$$

Therefore we find one of the solutions with the sequence *bcabd* from the construction.

**Definition 3.3.3.** A graph $G$ is *solvable* with respect to $\Sigma$, if and only if for every finite word in $\Sigma^*$ there exists a path in $G$ whose sequence of labels is a given word, i.e.,

$$G \text{ is solvable with respect to } \Sigma \Leftrightarrow \Delta(V, u) \neq \emptyset \text{ for all } u \in \Sigma^*.$$

This means a graph is not solvable if and only if an empty set is generated during the construction. Even and Meyer show that the problem of solving such an equation can be reduced to the problem of deciding whether a given graph is solvable with respect to some alphabet $\Sigma$. The problem of deciding whether $G$ is solvable is reduced to the problem of deciding whether $A$ accepts all words over $\Sigma$. A graph $G$ has a solution without delay if and only if there exists a nonempty set of vertices $V' \subseteq V$ such that $\Delta(\{v\}, l) \cap V' \neq \emptyset$ for all $v \in V'$ and $l \in \Sigma$.

**Example 3.3.4.** Let's consider a graph given in Figure 3.2, Table 3.3 represents this graph. We eliminate states $a$ and $c$ since $a$ has no transition with label 0 and $c$ has no outgoing edge with label 1, then $d$ has no transition with label 0 and $b$ is therefore eliminated. Hence $G$ has no solution with delay 0.

| | 0 | 1 |
|---|---|---|
| $b$ | $d$ | $b$ |
| $d$ | $-$ | $b$ |

Table 3.4: Eliminated table $T_2'$ of $G_2$

| | 0 | 1 |
|---|---|---|
| $b$ | $-$ | $b$ |

Table 3.5: Eliminated table $T_2''$ of $G_2$

## 3.4 Solvable graphs of sequential Boolean equations with delay $d$

**Definition 3.4.1.** A graph $G$ is *solvable with a finite delay* $d$ with respect to $\Sigma$ if the knowledge of first $d$ labels of a word $l_0 l_1 \dots l_{d-1}$ is sufficient to determine the initial vertex $v_0$, and for every $t$ the knowledge of $v_t, l_t l_{t+1} \dots l_{t+d}$ is sufficient to determine a vertex $v_{t+1}$, such that the determined part of the sequence of vertices is a part of the path with the given word of labels.

**Definition 3.4.2.** For a solution with a finite delay $d$ in $G$ the set of *graphs* $G^d$ for $d \geq 0$ is defined as follows:

1. $G^0 = G$;
2. The vertices of $G^d$ are $(d+1)$-tuples $(v, l_0, l_1, \dots, l_{d-1})$ where $v \in V$ and $l_0, l_1, \dots, l_{d-1} \in \Sigma$;
3. The vertex $(v, l_0, l_1, \dots, l_{d-1})$ is joined to vertex $(v', l_1, l_2, \dots, l_{d-1}, l)$ with the label $l$ if and only if $v' \in \delta(\{v\}, l_0)$;

4. There are no other edges in $G^d$.

Each vertex in $G^d$ contains information of the next $d$ letters. A transition in $G^d$ is equal to transitions of $d+1$ letters in $G$. To decide whether $G$ has a solution with delay $d$ is equivalent to decide whether $G^d$ has a solution with delay 1.

**Example 3.4.3.** We construct $G^1$ from Table 3.3 and represent the result in Table 3.6. We start with eliminating states $a0$ and $c1$, because they both have no outgoing edge with neither label 0 nor label 1. Through the same procedure all the states on Table 3.6 are eliminated, which means $G$ does not have a solution with delay 1 either. $G^2$ is constructed in Table 3.7. All states in $G^2$ are also eliminated. The same happens to $G^3$ in Table 3.8. Hence $G$ has a solution neither with delay 2 nor with delay 3.

| | 0 | 1 |
|---|---|---|
| $a0$ | – | – |
| $a1$ | $b0$ | $b1$ |
| $b0$ | $d0$ | $d1$ |
| $b1$ | $a0, b0, c0$ | $a0, b0, c0$ |
| $c0$ | $a0, b0$ | $a1, b1$ |
| $c1$ | – | – |
| $d0$ | $c0$ | $c1$ |
| $d1$ | $b0$ | $b1$ |

Table 3.6: Table of $G^1$

Hereto an algorithm for deciding whether or not a sequential Boolean equation has solution with delay $d$ is provided. Unfortunately we are not willing to go through the same procedure with any given delay $d$, but seek an efficient method for finding the upper bound of a finite-delay solution. In the following section, we introduce a method to determine the existence of a finite-delay solution for a sequential Boolean equation.

| | 0 | 1 |
|---|---|---|
| $a00$ | – | – |
| $a01$ | – | – |
| $a10$ | $b00$ | $b01$ |
| $a11$ | $b10$ | $b11$ |
| $b00$ | $d00$ | $d01$ |
| $b01$ | $d10$ | $d11$ |
| $b10$ | $a00, b00, c00$ | $a01, b01, c01$ |
| $b11$ | $a10, b10, c10$ | $a11, b11, c11$ |
| $c00$ | $a00, b00$ | $a01, b01$ |
| $c01$ | $a10, b10$ | $a11, b11$ |
| $c10$ | – | – |
| $c11$ | – | – |
| $d00$ | $c00$ | $c01$ |
| $d01$ | $c10$ | $c11$ |
| $d10$ | $b00$ | $b01$ |
| $d11$ | $b10$ | $b11$ |

Table 3.7: Table of $G^2$

## 3.5 Solvable graphs of sequential Boolean equations with finite delay

In the previous section we introduced a method to decide whether an automaton graph is solvable with a given delay $d$. Now we raise another problem of deciding whether an automaton graph is solvable with a finite delay. We look for an upper bound of a finite-delay solution in a graph $G$. In order to solve the problem we define a $d$-merge graph $G_d$.

**Definition 3.5.1.** Let $G = (V, V_0, E)$ be a graph with labels from $\Sigma$, a *$d$-merge of $G$*, denoted by $G_d$, is defined as follows:

1. The vertices of $G_d$ are $V$ from the graph $G$;
2. $\Sigma^d$ denotes the set of all label sequences of $\Sigma$ with length $d$;
3. Between $v_i \in V$ and $v_j \in V$ there is an edge in $G_d$ with label $l_0 l_1 \dots l_{d-1}$ for $l_0, l_1, \dots, l_{d-1} \in \Sigma$, if and only if a path from $v_i$ to $v_j$ in $G$ exists with the same label sequence.

| | 0 | 1 |
|---|---|---|
| $a000$ | – | – |
| $a001$ | – | – |
| $a010$ | – | – |
| $a011$ | – | – |
| $a100$ | $b000$ | $b001$ |
| $a101$ | $b010$ | $b011$ |
| $a110$ | $b100$ | $b101$ |
| $a111$ | $b110$ | $b111$ |
| $b000$ | $d000$ | $d001$ |
| $b001$ | $d010$ | $d011$ |
| $b010$ | $d100$ | $d101$ |
| $b011$ | $d110$ | $d111$ |
| $b100$ | $a000, b000, c000$ | $a001, b001, c001$ |
| $b101$ | $a010, b010, c010$ | $a011, b011, c011$ |
| $b110$ | $a100, b100, c100$ | $a101, b101, c101$ |
| $b111$ | $a110, b110, c110$ | $a111, b111, c111$ |
| $c000$ | $a000, b000$ | $a001, b001$ |
| $c001$ | $a010, b010$ | $a011, b011$ |
| $c010$ | $a100, b100$ | $a101, b101$ |
| $c011$ | $a110, b110$ | $a111, b111$ |
| $c100$ | – | – |
| $c101$ | – | – |
| $c110$ | – | – |
| $c111$ | – | – |
| $d000$ | $c000$ | $c001$ |
| $d001$ | $c010$ | $c011$ |
| $d010$ | $c100$ | $c101$ |
| $d011$ | $c110$ | $c111$ |
| $d100$ | $b000$ | $b001$ |
| $d101$ | $b010$ | $b011$ |
| $d110$ | $b100$ | $b101$ |
| $d111$ | $b110$ | $b111$ |

Table 3.8: Table of $G^3$

Even and Meyer declare that there exists an algorithm for deciding whether a given graph with $n$ vertices is solvable with a finite delay. It is proved by the following lemmata [6]:

**Lemma 3.5.2.** *If $G$ is solvable with delay $d$ then $G_d$ is solvable with delay 1.*

Since a label-alphabet on $G_k$ denoted by $\Sigma^k$ is the set of all label sequences of $\Sigma$ with length $k$, it is obvious that $d$ transitions in $G$ is equal to a transition in $G_k$.

**Lemma 3.5.3.** *If $G_d$ is solvable with delay 1 then $G$ is solvable with delay $2d-1$.*

Each $d$-merge graph $G_d$ which is solvable with delay 1 guarantees 2 transitions with each label sequence of length $d$ in $G$. Therefore the corresponding graph $G$ is solvable with delay $2d-1$.

**Definition 3.5.4.** Two graphs $G'$ and $G''$ are *similar* if they have the same set of vertices, i.e., $V' = V''$; and for every letter $l' \in \Sigma'$, with $\Sigma'$ denoting the set of labels of $G'$, there exists a letter $l'' \in \Sigma''$, with $\Sigma''$ denotes the set of labels of $G''$, such that for every $v \in V$, $\Delta'(\{v\}, l') = \Delta''(\{v\}, l'')$, and for every $l'' \in \Sigma''$ there exists an $l' \in \Sigma'$ for which the same condition holds.

Notice that the number of letters in $\Sigma'$ is not necessarily the same as the number of letters in $\Sigma''$.

**Lemma 3.5.5.** *If $G'$ and $G''$ are similar and $G'$ is solvable with delay $d$, then $G''$ is solvable with delay $d$.*

*Proof.* We define a relation between letters of $\Sigma'$ and letters of $\Sigma''$ for $l' \in \Sigma'$ and $l'' \in \Sigma''$ as follows:

$$l' \backsim l'' \Leftrightarrow \forall v \in V \ \Delta'(\{v\}, l') = \Delta''(\{v\}, l'').$$

This means for every $l' \in \Sigma'$ there exists an $l'' \in \Sigma''$, and for every $l'' \in \Sigma''$ there exists an $l' \in \Sigma'$, such that $l' \backsim l''$.

We assume that a word $l''_0 l''_1 \dots l''_{n-1}$ is given, $G''$ is solvable with delay 1 is demonstrated in the following steps:

1. Let the initial vertex be one of the initial vertices which is chosen in $G'$, if the first label on the word is $l'_0$ where $l'_0 \backsim l''_0$. Let $v_0$ be the initial vertex.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $a$ | $b, c$ | $a, c$ | $b, c$ | $a, b, c$ |
| $b$ | $c$ | $-$ | $-$ | $a$ |
| $c$ | $a$ | $-$ | $a, b$ | $b$ |

Table 3.9: Table of $G$

2. Let $l'_i$ be any letter of $\Sigma'$ which satisfies $l'_i \backsim l''_i$. For all $1 < i < n-1$ choose $v_i$ as the next vertex in $G'$, when the present vertex is $v_{i-1}$ and the first two letters are $l'_{i-1}$ and $l'_i$.

Because of the similarity, a safe transition for $G'$ implies a safe transition for $G''$. This lemma can be generalized to any delay $d$. □

**Lemma 3.5.6.** *If $G_i$ is similar to $G_j$ then, for every $k > 0$, $G_{i+k}$ is similar to $G_{j+k}$.*

*Proof.* Let $\Delta(\{v\}, l_0 l_1 \dots l_{j+h-1})$ be the set of vertices connected to $v$ with a path labeled $l_0 l_1 \dots l_{j+h-1}$ in $G_{j+h}$, therefore

$$\Delta(\{v\}, l_0 l_1 \dots l_{j+h-1}) = \Delta(\Delta(\{v\}, l_0 l_1 \dots l_{j-1}), l_j l_{j+1} \dots l_{j+h-1}).$$

Since $G_i$ is similar to $G_j$ there exists a word $l'_0 l'_1 \dots l'_{i-1}$ such that for all $v \in V$

$$\Delta(\{v\}, l_0 l_1 \dots l_{j-1}) = \Delta\left(\{v\}, l'_0 l'_1 \dots l'_{i-1}\right).$$

Thus,

$$\begin{aligned} \Delta(\{v\}, l_0 l_1 \dots l_{j+h-1}) &= \Delta\left(\Delta\left(\{v\}, l'_0 l'_1 \dots l'_{i-1}\right), l_j l_{j+1} \dots l_{j+h-1}\right) \\ &= \Delta\left(\{v\}, l'_0 l'_1 \dots l'_{i-1} l_j l_{j+1} \dots l_{j+h-1}\right). \end{aligned}$$

The same argument may be repeated with the roles of $i$ and $j$ reversed. □

**Example 3.5.7.** Let $G$ be a graph given in the Table 3.9.

With eliminating of label 3 we obtain a graph $G'$ represented in Table 3.10. According to the above-mentioned definition, $G$ is not solvable.

Table 3.11 represents a 2-merge graph $G_2$. $G'_2$ is obtained by elimination of the labels 00, 01, 02 and 11, 20, and represented in Table 3.12.

| | 0 | 1 | 2 |
|---|---|---|---|
| $a$ | $a, c$ | $a, c$ | $b, c$ |
| $b$ | $c$ | $-$ | $-$ |
| $c$ | $a$ | $-$ | $a, b$ |

Table 3.10: Table of $G'$

| | 00 | 01 | 02 | 10 | 11 | 12 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|
| $a$ | $a, c$ | $-$ | $a, b$ | $a, b, c$ | $a, c$ | $a, b, c$ | $a, c$ | $-$ | $a, b$ |
| $b$ | $a$ | $-$ | $a, b$ | $-$ | $-$ | $-$ | $-$ | $-$ | $-$ |
| $c$ | $b, c$ | $a, c$ | $b, c$ | $-$ | $-$ | $-$ | $b, c$ | $a, c$ | $b, c$ |

Table 3.11: Table of $G_2$

| | 01 | 11 | 22 |
|---|---|---|---|
| $a$ | $-$ | $a, c$ | $a, b$ |
| $b$ | $-$ | $-$ | $-$ |
| $c$ | $a, c$ | $-$ | $b, c$ |

Table 3.12: Table of $G'_2$

| | 010 | 011 | 012 | 110 | 111 | 112 | 220 | 221 | 222 |
|---|---|---|---|---|---|---|---|---|---|
| $a$ | $-$ | $-$ | $-$ | $a, b, c$ | $a, c$ | $a, b, c$ | $b, c$ | $a, c$ | $b, c$ |
| $b$ | $-$ | $-$ | $-$ | $-$ | $-$ | $-$ | $-$ | $-$ | $-$ |
| $c$ | $a, b, c$ | $a, c$ | $a, b, c$ | $-$ | $-$ | $-$ | $a, c$ | $-$ | $a, b$ |

Table 3.13: Table of $G_3$

| | 011 | 111 | 222 |
|---|---|---|---|
| $a$ | $-$ | $a, c$ | $b, c$ |
| $b$ | $-$ | $-$ | $-$ |
| $c$ | $a, c$ | $-$ | $a, b$ |

Table 3.14: Table of $G'_3$

| | 0110 | 0111 | 0112 | 1110 | 1111 | 1112 | 2220 | 2221 | 2222 |
|---|---|---|---|---|---|---|---|---|---|
| $a$ | – | – | – | $a, b, c$ | $a, c$ | $a, b, c$ | $a, c$ | – | $a, b$ |
| $b$ | – | – | – | – | – | – | – | – | – |
| $c$ | $a, b, c$ | $a, c$ | $a, b, c$ | – | – | – | $b, c$ | $a, c$ | $b, c$ |

Table 3.15: Table of $G_4$

| | 0111 | 1111 | 2222 |
|---|---|---|---|
| $a$ | – | $a, c$ | $a, b$ |
| $b$ | – | – | – |
| $c$ | $a, c$ | – | $b, c$ |

Table 3.16: Table of $G'_4$

$G'_2$ has no solution with delay 1. Therefore we construct $G_3$ with the words of length 3 consisting a label of $G'_2$ and a label of $G'$ as shown in Table 3.13.

010, 012, 110, 112, 220 and 221 are eliminated, therefore we obtain Table 3.14 representing the 3-merge graph $G'_3$.

We continue with the construction of 4-merge graph $G_4$, Table 3.15 represents $G_4$ which is constructed from the labels of $G'_3$ and $G'$.

The eliminated 4-merge graph $G'_4$ is represented in Table 3.16.

Since $G'_4$ is similar to $G'_2$ and $G'_4$ has no solution with delay 1, $k$-merge graph has no solution with delay 1 for all $k > 4$. Hence, $G$ has no solution with finite delay.

The algorithm of Even and Meyer provides algorithms solving not only the problem of deciding whether there exist solutions with finite delay for sequential Boolean equations, but also the problem of deciding whether player $O$ has finite-delay winning strategies in safety games. In Even-Meyer automaton graphs, we consider player $I$ choosing labels in a path as an input word, and player $O$ determining the corresponding output word by making moves according to the input bits from player $I$. A safety condition is fulfilled if there is always a safe transition in the automaton graph according to a input sequence from player $I$. Based on the result from Even and Meyer, Hosch and Landweber [8] proved that the problem whether Church's Problem with conditions stated in sequential calculus are solvable with delay is decidable.

# Chapter 4

# Solutions with a Fixed Delay in Infinite Games

We recalled in the previous chapter the algorithms from Even and Meyer [6] for solving sequential Boolean equations. In the Even-Meyer automaton graphs, player $I$ chooses input bits and player $O$ decides a path labeled with this input sequence. An automaton graph is solvable if and only if player $O$ always has a choice according to the input bits of player $I$. Therefore the Even-Meyer algorithm solves the problem of whether player $O$ has winning strategies with finite delay in safety games. We now consider whether there exists a $d$-delay solution for player $O$ in an arbitrary infinite game defined by a winning conditions $C(X, Y)$, i.e., whether the condition $C(X, Y)$ has a solution with a fixed delay $d$. In this chapter we develop algorithms over $d$-delay game graphs in arbitrary regular infinite games.

## 4.1 Game graphs with a fixed delay $d$

With the algorithm for solving sequential Boolean equations from Even and Meyer, Hosch and Landweber solved the problem of deciding whether Church's Problem with a condition $C(X, Y)$ stated in sequential calculus admits a finite automaton solution, and to provide the solution when it exists. Since game graphs offer more intuitive views, it's useful to study Church's Problem in the framework of infinite games. In order to solve Church's Problem with finite delay, we study algorithm solving the problem of deciding whether player $O$ has a finite-delay winning strategy in an infinite game specified by a condition $\varphi$ stated in sequential calculus.

To solve Church's Problem in the framework of infinite games, we distinguish the contribution of input bits from the set $\Sigma_I = \{0,1\}$ by player $I$ and output bits from the set $\Sigma_O = \{0,1\}$ player $O$. A specification $\varphi$ as the winning condition for player $O$ defines an infinite two-persons game. In this thesis, we consider a general game graph $G = (Q, \mu)$ as a 1-delay game graph, denoted by $G_1 = (Q_1, \mu_1)$, where $Q_1 = Q_{I_1} \dot{\cup} Q_{O_1}$. A transformation from an automaton graph $G_{\mathcal{A}}$ into a 1-delay game graph is shown in Figure 4.1. Let $|Q_{\mathcal{A}}|$ denote the size of automaton graph $G_{\mathcal{A}} = (Q_{\mathcal{A}}, \mu_{\mathcal{A}})$ and let $n = |Q_{\mathcal{A}}|$, the size of corresponding game graph $G_1 = (Q_1, \mu_1)$ is equal to $3n$, i.e., $|Q_1| = 3n$.

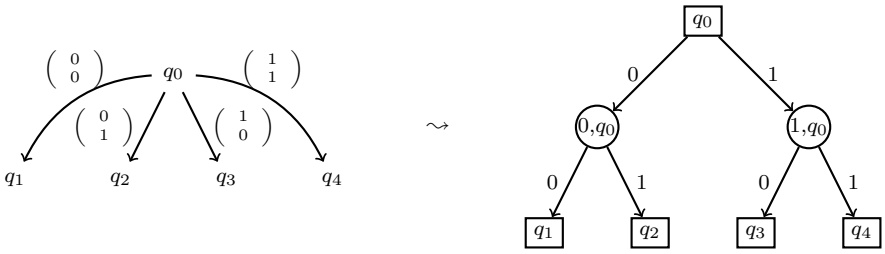


Figure 4.1: Transformation from automaton graph $G_{\mathcal{A}}$ to 1-delay game graph $G_1$

**Definition 4.1.1.** Let $G_1 = (Q_1, \mu_1)$ be a 1-delay game graph and $I_d \subseteq \{0,1\}^d$ be the set of input sequences with length $d$. A *d-delay game graph* is a tuple $G_d = (Q_d, \mu_d)$ with $Q_d = Q_{I_d} \dot{\cup} Q_{O_d}$, where $Q_{I_d} = I_{d-1} \times Q_1$ denotes the set of vertices for player $I$ in $G_d$, $Q_{O_d} = I_d \times Q_1$ denotes the set of vertices for player $O$ in $G_d$, and

$$\mu := \{Q_{I_d} \times \Sigma_I \to Q_{O_d}\} \cup \{Q_{O_d} \times \Sigma_O \to Q_{I_d}\}$$

is the function determining the moves. The corresponding edge relation $E_d \subseteq \{Q_{I_d} \times Q_{O_d}\} \cup \{Q_{O_d} \times Q_{I_d}\}$ is defined as $(q,p) \in E_d$ if and only if $\mu(q,u) = p$ for some $u \in \{0,1\}$.

Note that all outgoing edges are connected only with vertices of the opponent. The vertices in the set $Q_{I_d}$ are depicted by boxes and the vertices in the set $Q_{O_d}$ by circles in $d$-delay game graphs.

**Definition 4.1.2.** A *play in* $G_d$ is a sequence

$$\rho = (i_o \dots i_{d-2}, q_0)\,(i_o \dots i_{d-1}, q_0)\,(i_1 \dots i_{d-1}, q_1)\,(i_1 \dots i_d, q_1) \dots$$

with

- $((i_j \dots i_{j+d-2}, q_j), (i_j \dots i_{j+d-1}, q_j)) \in E_d$;
- $((i_j \dots i_{j+d-1}, q_j), (i_{j+1} \dots i_{j+d-1}, q_{j+1})) \in E_d$.

Let $I_d \subseteq \{0,1\}^d$ be the set of input sequences with length $d$. For $d > 0$ and $q_k \in Q_1$, the edges in $G_d = (Q_d, \mu_d)$ are labeled by input bits $i_{k+d-1}$ from vertices in $I_{d-1} \times \{q_k\}$ for player $I$, and output bits $o_k$ from vertices in $I_d \times \{q_k\}$ for player $O$ respectively.

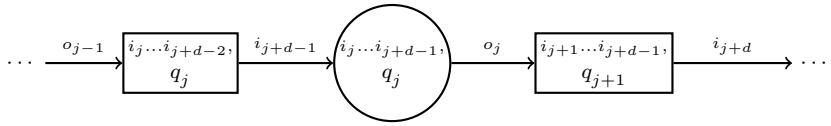


Figure 4.2: Subgraph of a $d$-delay game graph $G_d$

All vertices in $I_{d-1} \times \{q_k\}$ for player $I$ and in $I_d \times \{q_k\}$ for player $O$ are final vertices in $d$-delay game graph $G_d$ if $q_k$ is a final state in the corresponding automaton graph $G_{\mathcal{A}}$.

**Definition 4.1.3.** An *infinite game with delay* $d$ is a tuple $(G_d, I_{d-1} \times \{q_0\}, \varphi)$ consisting of a game graph $G_d = (Q_d, \mu_d)$, the set of initial vertices $I_{d-1} \times \{q_0\}$ with $q_0$ is the initial state in the corresponding automaton graph, and a winning condition $\varphi$ for player $O$. Player $O$ wins a play $\rho$ if $\rho \in Q_d^\omega$ fulfills $\varphi$.

A $d$-delay strategy for player $O$ starting from a vertex in $I_{d-1} \times \{q_0\}$ in a $d$-delay game graph $G_d$ is defined analogously as in a general game graph $G$, i.e., it is a function $f_d : Q_d \to \{0,1\}$. Let $\mathrm{Win} \subseteq Q_d^\omega$ be the set of the plays won by player $O$, a $d$-delay strategy $f_d$ for player $O$ from a vertex in $I_{d-1} \times \{q_0\}$ is called a $d$-delay winning strategy from an initial vertex if and only if every play $\rho$ consistent with $f_d$ is in Win.

**Lemma 4.1.4.** *Player $O$ has a $d$-delay winning strategy from vertex $q_k \in Q_1$ in an infinite game if and only if player $O$ has a winning strategy from all vertices in $I_{d-1} \times \{q_k\}$ in the corresponding $d$-delay game graph $G_d$.*

For solving Church's Problem, player $O$ wins an infinite game with delay $d$ if and only if player $O$ has a winning strategy from each initial vertex in $I_{d-1} \times \{q_0\}$ in $G_d$.

## 4.2 Strategies with a fixed delay in reachability games

In this section we study the algorithm for determining a solution with a fixed delay $d$ in a reachability game. In the corresponding $d$-delay reachability game graph we construct an attractor winning strategy and the winning region of player $O$, such that we determine whether or not player $O$ has a $d$-delay winning strategy in the reachability game.

**Definition 4.2.1.** Let $G = (Q, \mu)$ be a game graph and $F \subseteq Q$, the *reachability winning condition* is

$$\rho \in \mathrm{Win} :\Leftrightarrow \exists i : \rho(i) \in F.$$

In reachability games, the winning regions $W_I$ of player $I$ and $W_O$ of player $O$ can be computed in polynomial time as well as corresponding positional winning strategies - attractor strategies. For $i = 0, 1, 2, \ldots$ construct the sets $\mathrm{Attr}_O^i(F)$ where

$$\mathrm{Attr}_O^i(F) = \{q \in Q \mid \text{from } q \text{ player } O \text{ can force a visit to } F \text{ in } \leq i \text{ moves}\}.$$

Inductive construction of $\mathrm{Attr}_O^i(F)$ over $i$ is shown as follows:

$$\begin{aligned} \mathrm{Attr}_O^0(F) &= F; \\ \mathrm{Attr}_O^{i+1}(F) &= \mathrm{Attr}_O^i(F) \\ &\cup \left\{q \in Q_O \mid \exists (q,p) \in E : p \in \mathrm{Attr}_O^i(F)\right\} \\ &\cup \left\{q \in Q_I \mid \forall p \in Q, (q,p) \in E \rightarrow p \in \mathrm{Attr}_O^i(F)\right\}. \end{aligned}$$

Obviously $\mathrm{Attr}_O^0(F) \subseteq \mathrm{Attr}_O^1(F) \subseteq \mathrm{Attr}_O^2(F) \subseteq \ldots \subseteq \mathrm{Attr}_O^{|Q|}(F) = \mathrm{Attr}_O^{\omega}(F)$ holds.

**Definition 4.2.2.** The *attractor strategy* of player $O$ is defined as follows: from a vertex in $\mathrm{Attr}_O^{i+1}(F) \setminus \mathrm{Attr}_O^i(F)$ go to a vertex in $\mathrm{Attr}_O^i(F)$.

With the following algorithm we can determine the winning regions $W_I$ of player $I$ and $W_O$ of player $O$ in a given reachability game $G = (Q, \mu)$ and $F \subseteq Q$:

1. Compute outgoing edges $out(q)$ for each vertex $q \in Q_I$;
2. Set $n(q) := out(q)$ for each $q \in Q_I$;
3. Perform breadth-first search backwards from $F$ as follows:
   - Mark all $q \in F$,
   - Mark $q \in Q_O$ if $q$ reaches marked vertex with an outgoing edge,
   - For $q \in Q_I$ reached backwards from marked vertex set $n(q) := n(q) - 1$, and mark $q \in Q_I$ when $n(q) = 0$,
   - The marked vertices are in $\mathrm{Attr}_O^{|Q|}(F)$.

For the reachability condition, if the state $q_k$ in $G_{\mathcal{A}}$ is a final state, then the vertex $q_k$ for player $I$ and all vertices in $\{0,1\} \times \{q_k\}$ for player $O$ are the final vertices in $G_1$.

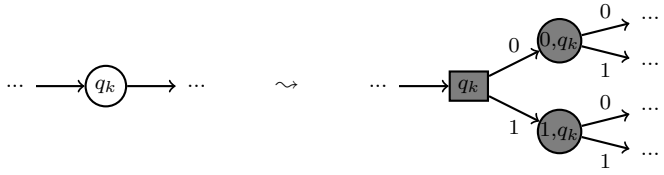


Figure 4.3: Transformation of a final state in $G_{\mathcal{A}}$ into final vertices in $G_1$

**Definition 4.2.3.** A *d-delay attractor strategy* in a reachability game is defined by an attractor strategy which is constructed in the corresponding $d$-delay reachability game graph $G_d$.

Let $F_d$ be the set of the final vertices in $G_d$, a $d$-delay attractor strategy for player $O$ starting at a vertex in the set of $I_{d-1} \times q_0$ is defined as follows: from a vertex in $\mathrm{Attr}_O^{i+1}(F_d) \setminus \mathrm{Attr}_O^{i}(F_d)$ go to a vertex in $\mathrm{Attr}_O^{i}(F_d)$.

**Lemma 4.2.4.** *Player $O$ has a $d$-delay attractor strategy from $q_0$ in a 1-delay reachability game graph $G_1$ if and only if there exists an attractor winning strategy for player $O$ from all vertices in $I_{d-1} \times \{q_0\}$ in $G_d$.*

In other words, player $O$ wins a reachability game with delay $d$ if and only if each vertex in $I_{d-1} \times \{q_0\}$ belongs to $\mathrm{Attr}_O^{|Q_d|}(F_d)$ in $d$-delay game graph $G_d$.

**Example 4.2.5.** We consider the following reachability game with a simple regular winning condition for player $O$: the output bit of player $O$ at time 0 is equal to the input bit of player $I$ at time 1; afterwards player $I$ can decide input bits arbitrarily, and analogously for player $O$ to decide output bits, i.e.,

$$\left[\begin{pmatrix} * \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ * \end{pmatrix} + \begin{pmatrix} * \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ * \end{pmatrix}\right]\begin{pmatrix} * \\ * \end{pmatrix}^{\omega}.$$

The symbol $*$ denotes an arbitrary letter from $\{0,1\}$. We construct in Figure 4.4 the automaton graph $G_{\mathcal{A}}$ according to the above-mentioned winning condition for player $O$.

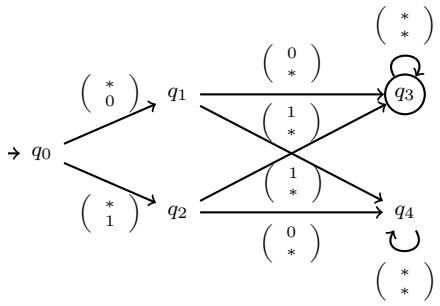


Figure 4.4: An automaton graph $G_{\mathcal{A}}$ with reachability condition

The 1-delay reachability game graph $G_1$ is constructed from the automaton graph $G_{\mathcal{A}}$ in Figure 4.5 with $F_1 = \{q_3, \{0,1\} \times \{q_3\}\}$.

An attractor strategy for player $O$ is denoted by '$\twoheadrightarrow$' in $G_1$. We construct the winning region $W_O$ of player $O$ in $G_1$. Player $O$ does not win the reachability game with 1-delay attractor strategy since $q_0 \notin \mathrm{Attr}_O^{|Q_1|}(F_1)$. Hence, we continue with the construction of the 2-delay game graph $G_2$.

In Figure 4.6 the vertices in $\mathrm{Attr}_O^{|Q_2|}(F_2)$ are gray. Since all initial vertices are in $W_O$, player $O$ has an attractor strategy from each vertex in $\{0,1\} \times \{q_0\}$ in $G_2$. This means player $O$ has a 2-delay attractor winning strategy in the reachability game.

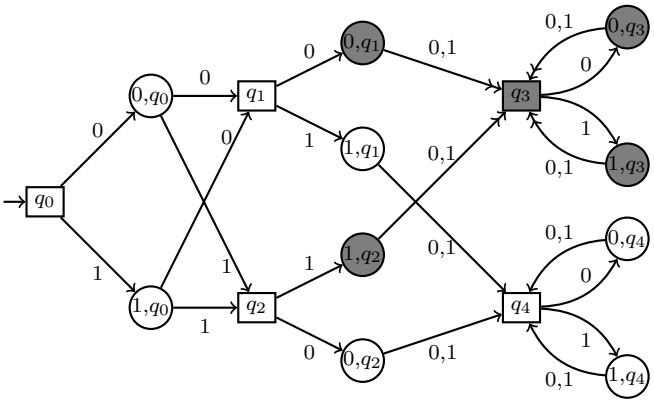


Figure 4.5: 1-delay game graph $G_1$ of a reachability game

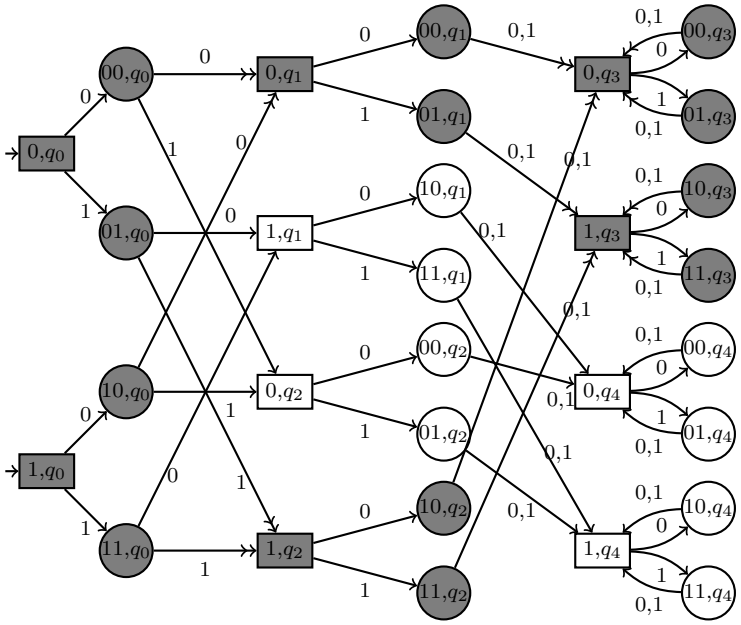


Figure 4.6: 2-delay game graph $G_2$ of a reachability game

**Definition 4.2.6.** Let $G = (Q, \mu)$ be a game graph, the *safety winning condition* is

$$\rho \in Win \Leftrightarrow \forall i \; \rho(i) \in F.$$

We consider a reachability game over $G$ with the winning condition $\rho \in \mathrm{Win}' \Leftrightarrow \exists i \rho(i) \in Q \backslash F$. Since reachability games and safety games are dual, the winning region $W_I'$ of player $I$ is the winning region $W_O$ of player $O$ in the safety game.

## 4.3 Strategies with a fixed delay in weak parity games

In this section we study the algorithm for determining determining a solution with a fixed delay $d$ in a weak parity game. Reachability games are considered as a special case of weak parity games.

**Definition 4.3.1.** A *Staiger-Wagner game* is also called obligation game, for $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$ with $F_i \subseteq Q$ its winning condition is

$$\rho \in \mathrm{Win} :\Leftrightarrow \mathrm{Occ}(\rho) \in \mathcal{F}.$$

Weak parity games is a special case of Staiger-Wagner games.

**Definition 4.3.2.** Let $G = (Q, \mu)$ be a game graph and $c$ be a coloring function $c : Q \to \{0, \ldots, k\}$, a play $\rho = \rho(0)\rho(1)\rho(2)\ldots$ delivers a sequence of colors $c(\rho) = c(\rho(0))\, c(\rho(1))\, c(\rho(2))\ldots$ The *weak parity condition* defines the set of plays won by player $O$ by

$$\rho \in \mathrm{Win} :\Leftrightarrow \max(\mathrm{Occ}(c(\rho))) \text{ is even;}$$

i.e., the maximal color occurring in $\rho$ is even. An infinite game with such a weak parity condition is called *weak parity game.*

**Theorem 4.3.3** ([16])**.** *In weak parity games the winning regions $W_I$ and $W_O$ with $W_I \dot{\cup} W_O = Q$ can be computed, and the corresponding positional winning strategies can be determined.*

Let $G = (Q, \mu)$, a coloring function $c : Q \to \{0, 1, \ldots, k\}$ where k is even, set $C_i := \{q \in Q \mid c(q) = i\}$. We calculate the sets $A_k, A_{k-1}, \ldots, A_0$:

$$\begin{aligned} A_k &:= \mathrm{Attr}_O(C_k); \\ A_i &:= \begin{cases} \mathrm{Attr}_O(C_i \backslash (A_{i+1} \cup \ldots \cup A_k)) & \text{if } i \text{ even} \\ \mathrm{Attr}_I(C_i \backslash (A_{i+1} \cup \ldots \cup A_k)) & \text{if } i \text{ odd} \end{cases} \quad \text{for } i \leq k-1. \end{aligned}$$

Each vertex is in one of the sets $A_0, \ldots, A_k$, therefore

$$W_I = \bigcup_{i \text{ odd}} A_i \text{ and } W_O = \bigcup_{i \text{ even}} A_i;$$

i.e., the union of the associated attractor strategies are positional winning strategies for player $I$ on $W_I$ and player $O$ on $W_O$ respectively.

A reachability game over $G$ with $\rho \in \text{Win} \Leftrightarrow \exists i \ \rho(i) \in F$ can be described as a special weak parity game over $G$ with coloring

$$c(q) = \begin{cases} 2, & \text{for} \quad q \in F \\ 1, & \text{for} \quad q \notin F \end{cases}$$

A safety game over $G$ with $\rho \in \text{Win} \Leftrightarrow \forall i \ \rho(i) \in F$ can be described as a special weak parity game over $G$ with coloring

$$c(q) = \begin{cases} 0, & \text{for} \quad q \in F \\ 1, & \text{for} \quad q \notin F \end{cases}$$

We construct a $d$-delay game graph for a weak parity game. Let $G_1 = (Q_1, \mu_1)$ be a 1-delay weak parity game graph, which is constructed from an automaton graph $G_{\mathcal{A}}$. The corresponding $d$-delay game graph $G_d = (Q_d, \mu_d)$ for $d > 0$ consists of a set of vertices in $c(q_k) \times I_{d-1} \times \{q_k\}$ for player $I$, from which the outgoing edges are labeled by input bits $i_{k+d-1}$; and a set of vertices in $c(q_k) \times I_d \times \{q_k\}$ for player $O$, from which the outgoing edges are labeled by output bits $o_k$. The vertices for player $I$ are depicted by boxes and the vertices for player $O$ are denoted by circles in $d$-delay game graphs. With the algorithm for positional winning strategies in weak parity games recalled above, we can construct winning regions $W_I$ and $W_O$ of player $I$ and player $O$ respectively in weak parity game graph $G_d$, and the corresponding $d$-delay winning strategies for both players in their winning regions.

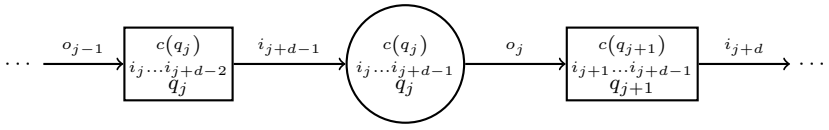


Figure 4.7: Subgraph of a $d$-delay game graph $G_d$ of a weak parity game

The next lemma follows from the construction presented above.

**Lemma 4.3.4.** *Player $O$ has a $d$-delay positional winning strategy from the initial vertex $q_0$ in a weak parity game if and only if there exists a positional winning strategy for player $O$ from each initial vertex in $I_{d-1} \times \{q_0\}$ in $G_d$.*

**Example 4.3.5.** We color the vertices in the game graph from Example 5.2.1 in Figure 4.7. With the above-mentioned algorithm we can construct the positional winning strategy and the winning region $W_O$ of player $O$ in the $d$-delay game graph of this weak parity game.

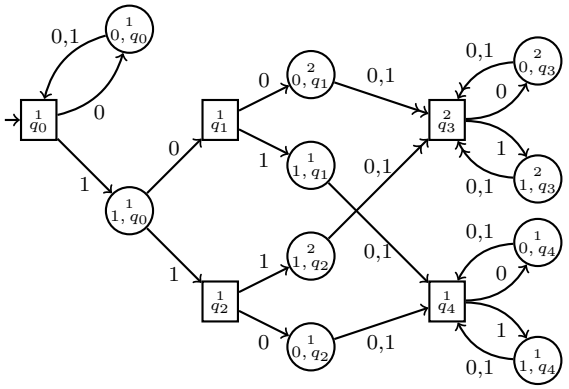


Figure 4.8: 1-delay game graph $G_1$ of the weak parity game from Example 5.1.2

## 4.4 Strategies with a fixed delay in Büchi games

**Definition 4.4.1.** Let $G = (Q, \mu)$ be a graph with $Q_I \dot{\cup} Q_O = Q$ and $F \subseteq Q$, a game $(G, q_0, \varphi)$ with an initial vertex $q_0$ and the winning conditions $\varphi$ for player $O$

$$\varphi : \rho \in \text{Win} :\Leftrightarrow \text{Inf}\,(\rho \cap F \neq \emptyset)$$

is called a *Büchi game*, i.e., player $O$ wins a play *rho* in a Büchi game if and only if the play $\rho$ visits $F$ infinitely often.

Let $Recur_O^i\,(F)$ for $i \geq 1$ denotes the set of vertices $q \in F$ from which player $O$ can force at least $i$ revisits to $F$, i.e.,

$$\begin{aligned} Recur_O^0(F) &:= F; \\ Recur_O^i(F) &:= F \cap \mathrm{Attr}_O^{|Q|}\left(Recur_O^{i-1}(F)\right); \\ Recur_O(F) &:= \bigcap_{i\geq 0} Recur_O^i(F). \end{aligned}$$

Thus $F \supseteq Recur_O^1(F) \supseteq Recur_O^2(F) \supseteq \ldots$ holds. The winning region of player $O$ is constructed by $W_O = \mathrm{Attr}_O^{|Q|}(Recur_O(F))$. The positional winning strategy with delay $d$ and the corresponding winning region of player $O$ in a Büchi game can be determined in a $d$-delay game graph $G_d$ for the Büchi game according to the above-mentioned algorithm. If all initial vertices in $I_{d-1} \times \{q_0\}$ in a $d$-delay Büchi game graph $G_d$ belong to $W_O$, player $O$ wins the Büchi game with delay $d$, which leads to the following lemma.

**Lemma 4.4.2.** *Player $O$ has a $d$-delay winning strategy from the initial vertex $q_0$ in a Büchi game if and only if there exists a winning strategy for player $O$ from each initial vertex in $I_{d-1} \times \{q_0\}$ in $G_d$.*

## 4.5 Strategies with a fixed delay in parity games

**Definition 4.5.1.** Given a game graph $G = (Q, \mu)$, a *parity game* with a coloring function $c : Q \to \{0, 1, \ldots, k\}$ where $k$ is even, and $C_i := \{q \in Q \mid c(q) = i\}$, is defined by the winning condition

$$\varphi : \rho \in \mathrm{Win} \Leftrightarrow \max(\mathrm{Inf}(c(\rho))) \text{ even}.$$

**Theorem 4.5.2.** *Let $G = (Q, \mu)$ be a game graph with a coloring function $c : Q \to \{0, 1, \ldots, k\}$ and parity winning condition, then*

1. *Every vertex in $Q$ belongs to either $W_I$ or $W_O$, i.e., parity games are determined;*
2. *If $G$ is finite, the winning regions $W_I$ and $W_O$ with $Q_I \dot{\cup} Q_O = Q$ and the corresponding positional winning strategies for player $I$ and player $O$ can be determined.*

The positional winning strategy with delay $d$ for player $O$ in a parity game can be determined in a $d$-delay parity game graph $G_d$ analogously, according to the algorithms mentioned in the previous sections. If all initial vertices $I_{d-1} \times \{q_0\}$ of a $d$-delay parity game graph belong to $W_O$, player $O$ wins the parity game with delay $d$, as formulated in the following lemma.

**Lemma 4.5.3.** *Player $O$ has a $d$-delay positional winning strategy from the initial vertex $q_0$ in a parity game if and only if there exists a winning strategy for player $O$ from each initial vertex in $I_{d-1} \times \{q_0\}$ in $G_d$.*

# Chapter 5

# Solutions with Bounded Delay in Infinite Games

In the previous chapter we have shown how to determine whether there exists a winning strategy with a fixed delay $d$ for player $O$ in an infinite game. We constructed finite-state $d$-delay game graphs $G_d$ for an infinite game for $d > 0$. In these infinite games we compute positional winning strategies with delay and defined the winning regions of player $O$. Now we consider the problem whether or not player $O$ has a winning strategy with a bounded delay in an infinite game; and if not, how can we confirm the assumption and when shall we stop the construction of $G_d$ (i.e., the breaking condition). We present algorithms for solving this problem in the following sections.

## 5.1 Absorbed game graphs

The function representing the size of the winning region of player $O$ in a certain game depending on the delay $d$ is monotone, but not necessarily strictly monotone. We illustrate this fact by constructing absorbed game graphs, which build a bridge between the game graph $G_1$ and the $d$-delay game graph $G_d$. It allows to compare the winning regions of player $O$ in game graphs with different finite delay.

**Definition 5.1.1.** Given a 1-delay game graph $G_1 = (Q_1, \mu_1)$, a $d$-delay absorbed game graph $G^d = (Q_1, \mu_1, l)$ consists of the same vertices and the same edges as in $G_1$, and a labeling function $l : Q_1 \to \{1, \ldots, d, \infty\}$. A 1-delay absorbed game graph $G^1$ is initialized by marking all vertices by $\infty$.

A *d-delay absorbed game graph* $G^d$ is constructed based on the $d-1$-delay absorbed game graph $G^{d-1}$ for $d > 0$ as follows:

- A vertex in $q \in Q_{O_1}$ for player $O$ is marked by $d$ if and only if the vertex in $q$ in $G^{d-1}$ is marked by $\infty$, and all vertices in $I_d \times \{q\}$ are in $W_O$ in $G_d$, i.e., each vertex in $I_d \times \{q\}$ has a successor which belongs to $W_O$ in $G_d$;
- A vertex $q \in Q_{I_1}$ for player $I$ is marked by $d$ if and only if $q$ in $G^{d-1}$ is marked by $\infty$, and all vertices in $I_{d-1} \times \{q\}$ are in $W_O$ in $G_d$, i.e., all successors of each vertex in $I_{d-1} \times \{q\}$ belong to $W_O$ in $G_d$;
- The rest of vertices retain the same label as in $G^{d-1}$.

Notice that the vertices marked by a element from $\mathbb{N}$ in $G^{d-1}$ will not be re-marked in $G^d$. Since player $O$ wins an infinite game from a vertex $q \in Q_1$ with delay $d'$ for all $d' \geq d$, if the vertex $q$ is marked in a $d$-delay absorbed game graph $G^d$ with $d$. The winning region $W_O^d$ of player $O$ in $G^d$ consists of the vertices marked by $n$ for $0 < n \leq d$. Therefore the winning region $W_I^d$ of player $I$ in $G^d$ consists of the vertices marked by $\infty$.

Obviously $W_O^1 \subseteq W_O^2 \subseteq W_O^3 \subseteq \ldots$ holds. According to the construction of absorbed game graphs from $d$-delay game graphs, we construct the corresponding $d$-delay absorbed game graph for the reachability game in the example from the previous chapter.

**Example 5.1.2.** Let us look backwards at Example 4.2.5. The corresponding 1-delay absorbed game graph $G^1$ is shown in Figure 5.1 and 2-delay absorbed game graph $G^2$ is shown in Figure 5.2.

In this reachability game we have $W_O^1 \subset W_O^2 = W_O^3 = \ldots$, since player $O$ wins the reachability game from the initial vertex $q_0$ with delay 2.

**Lemma 5.1.3.** *Player $O$ wins an infinite game with a finite delay $d$ from vertex $q \in Q$ if and only if vertex $q$ in the corresponding absorbed game graph $G^d$ is labeled by $d$.*

We know that the winning region $W_O^d$ of player $O$ in $G^d$ for $d > 0$ increases monotonically with the increase of $d$, i.e., $W_O^1 \subseteq W_O^2 \subseteq W_O^3 \subseteq \ldots$, but Example 5.1.4 for another reachability game shows us that the increase of $W_O$ is not strict.

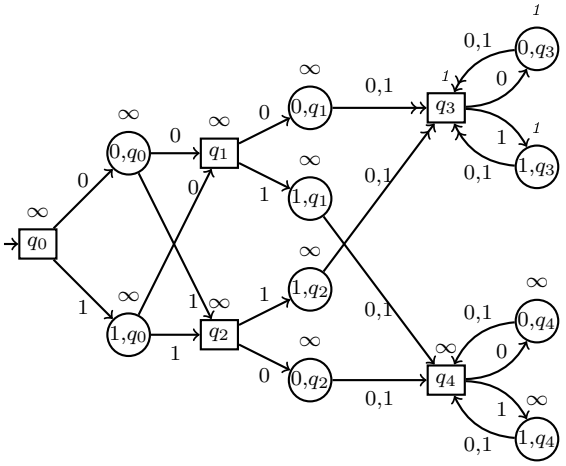


Figure 5.1: 1-delay absorbed game graph $G^1$

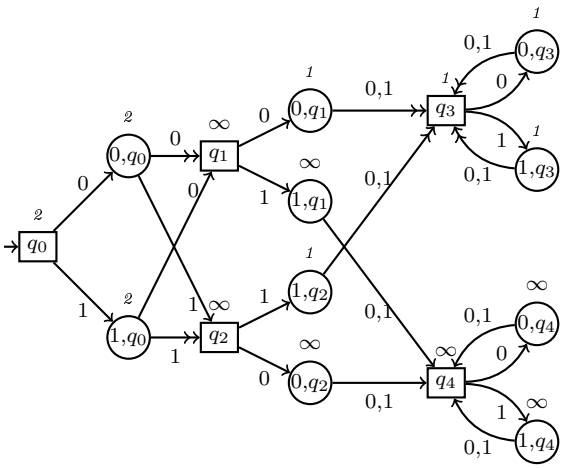


Figure 5.2: 2-delay absorbed game graph $G^2$

**Example 5.1.4.** We consider the following reachability game with a regular winning condition for player $O$: if the input of player $I$ at time 0 is equal to 0, then the output of player $O$ at time 0 have to be same as the input of player $I$ in 1 time unit later; and if the input at time 0 is 1, the output of

player $O$ at time 0 should be equal to the output of player $O$ at time 3, i.e.,

$$\left[\left[\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0\\*\end{pmatrix}+\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1\\*\end{pmatrix}\right]+\right.$$
$$\left.\left[\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}*\\*\end{pmatrix}\begin{pmatrix}*\\*\end{pmatrix}\begin{pmatrix}0\\*\end{pmatrix}+\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}*\\*\end{pmatrix}\begin{pmatrix}*\\*\end{pmatrix}\begin{pmatrix}1\\*\end{pmatrix}\right]\right]\begin{pmatrix}*\\*\end{pmatrix}^{\omega}.$$

The symbol $*$ denotes an arbitrary letter from $\{0,1\}$. The corresponding automaton graph $G_{\mathcal{A}}$ is constructed in Figure 5.3. It is obvious that player $O$ wins the reachability game with a 4-delay attractor strategy. The corresponding 4-delay absorbed game graph $G^4$ is constructed in Figure 5.4.

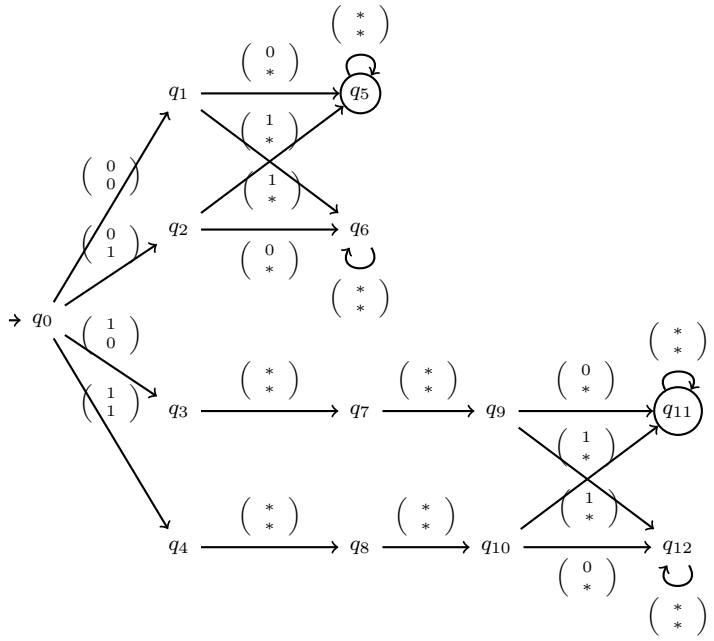


Figure 5.3: Automaton graph $G_{\mathcal{A}}$ with reachability condition

In this reachability game we have $W_O^1 \subset W_O^2 = W_O^3 \subset W_O^4 = W_O^5 = \ldots$.

**Definition 5.1.5.** Player $O$ wins an infinite game from the initial vertex $q_0 \in Q_1$ over $G_1 = (Q_1, \mu_1)$ with *bounded delay*, if there exists a $d$ for $d > 0$ such that player $O$ wins the infinite game from $q$ with delay $d$.

In the following sections we study how to solve the problem of whether a bounded-delay solution in an infinite game exists.

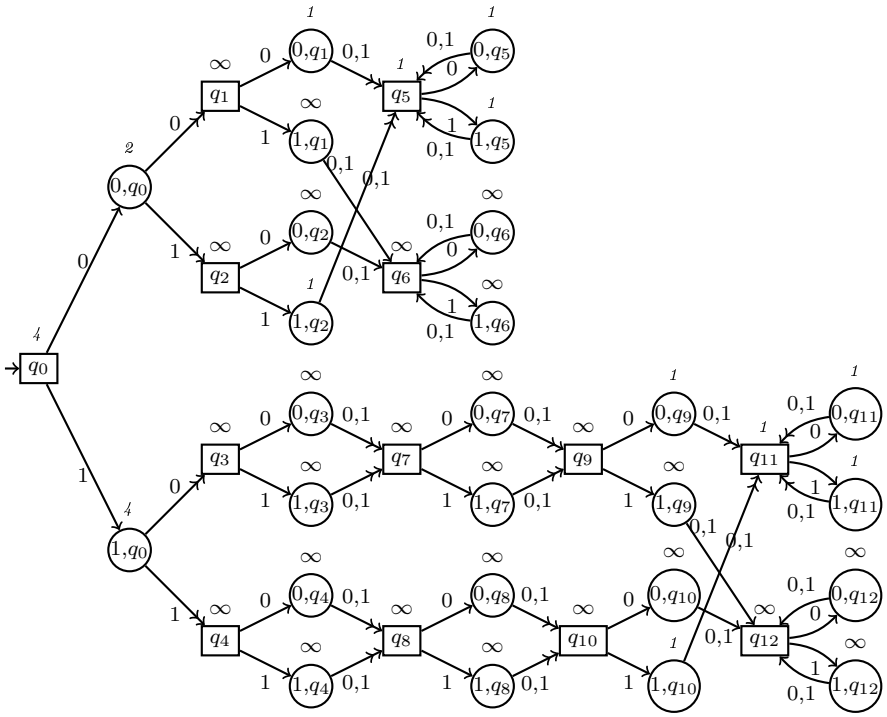


Figure 5.4: 4-delay absorbed game graph $G^4$

## 5.2 Winning strategies with bounded delay in reachability games

Since the winning region of player $O$ in the absorbed game graphs $G^d$ for $d > 0$ increases monotonically with the increase of delay $d$, but not strictly monotonically, a problem to decide the breaking condition of the construction of $d$-delay game graph $G_d$ is raised. In this section we introduce an algorithm to determine a breaking condition of the construction of $d$-delay game graph, i.e., to determine the existence of a winning strategy with a bounded delay for player $O$ in an infinite game.

The following example shows a reachability game which is not solvable with a finite delay.

**Example 5.2.1.** We now consider a reachability game with a regular winning condition for player $O$: the equality between the output of player $O$ at time

$t$ and the input of player $I$ at time $t+1$ is required, but only when the input of player $I$ at time $t$ is 1, i.e.,

$$\begin{pmatrix} 0 \\ * \end{pmatrix}^* \left[ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ * \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ * \end{pmatrix} \right] \begin{pmatrix} * \\ * \end{pmatrix}^\omega .$$

We construct the corresponding automaton graph $G_{\mathcal{A}}$ in Figure 5.5.

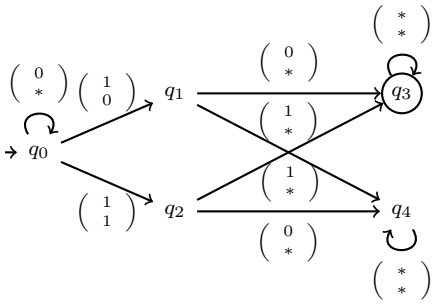


Figure 5.5: Automaton graph $G_{\mathcal{A}}$ with reachability condition

We construct the 1-delay game graph $G_1$ of the reachability game from the automaton graph $G_{\mathcal{A}}$ in Figure 5.6 with $F_1 = \{q_3, \{0,1\} \times \{q_3\}\}$.

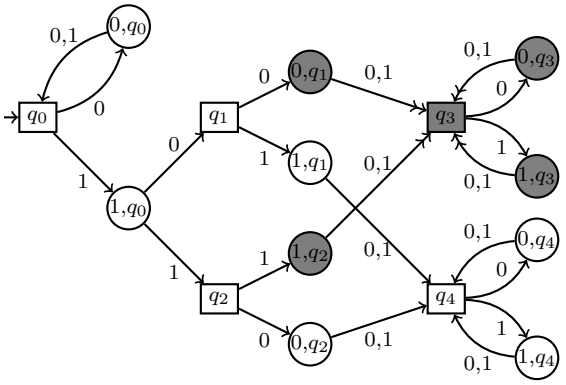


Figure 5.6: 1-delay game graph $G_1$

An attractor strategy for player $O$ denoted by '↠' in $G_1$, and the vertices in the winning region $W_O = \mathrm{Attr}_O^{|Q_1|}(F_1)$ of player $O$ are colored in gray in Figure Figure 5.6. But player $O$ has no attractor winning strategy from the

initial vertex $q_0$ to reach $F$ in the 1-delay game graph $G_1$, we continue with the construction of game graph $G_2$, which is represented in Figure 5.7.

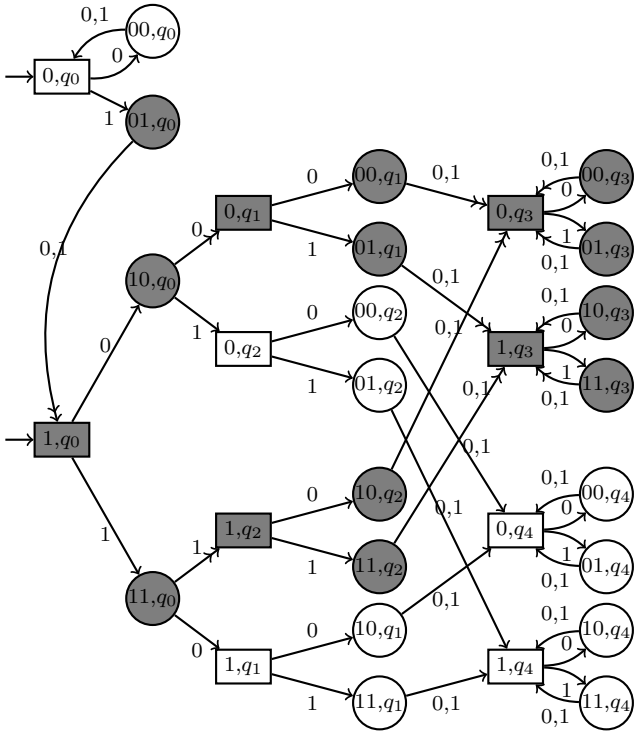


Figure 5.7: 2-delay game graph $G_2$

Since $W_O^1 = W_O^2 = W_O^3 = \ldots$ holds in this reachability game, player $O$ has no attractor winning strategy with any finite delay. The vertex $\left(0^{d-1}, q_0\right) \notin \mathrm{Attr}_O^{|Q_d|}(F_d)$ for all $d > 0$. We need to decide the breaking condition for the construction of $d$-delay game graphs $G_d$ for $d > 0$ in the general case. For this goal we need to decide whether there exists an attractor winning strategy for player $O$ with a bounded delay $d$.

Since $d$-delay game graphs for an infinite game grow exponentially with the increase of delay $d$ for $d > 0$, we can hardly find a simple way to determine a breaking condition in such $d$-delay game graphs. Thus we turn now to another aspect of understanding a $d$-delay game graph, and consider the problem in the aspect of unfolding trees consisting of all plays in an infinite game.

**Remark 5.2.2.** The plays in a game graph $G_1$ with delay $d$ from vertex $q \in Q_1$ can be seen as a binary tree from root $q$ with depth $2d$, we call it the *$d$-delay tree from $q$* and denote by $T_d$.

**Definition 5.2.3.** We reconstruct $T_d$ into *$d$-delay directed acyclic graphs* from $q \in Q_1$ denoted by $DAG_d$ by merging all doubling vertices on the same level of tree $T_d$ as follows:

1. On each level of the $d$-delay tree all doubling vertices are merged, hence in the corresponding $DAG_d$ all levels contain only identifiable vertices which first occur on each level in $T_d$, e.g. uppermost on our picture;

2. All incoming edges from the predecessors of merged vertices are therefore connected to the same vertex on the same level in $DAG_d$.

Since the game graph $G_1$ is finite, the width of $DAG_d$ are eventually limited by the size of game graph $|Q_1|$.

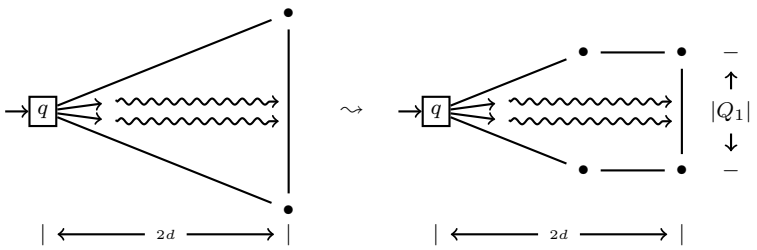


Figure 5.8: Transformation from $d$-delay tree $T_d$ into $DAG_d$ with root $q_j \in Q_1$

**Definition 5.2.4.** An *$I_d$-determined subgraph $T_{I_d}$ from $q \in Q_1$* is a subgraph of a $d$-delay tree $T_d$ with root $q$ in which the next $d$ moves of player $I$ are determined, i.e., each vertex for player $I$ on the same odd level in $T_{I_d}$ has only one outgoing edge with the same label as shown in Figure 5.9.

**Definition 5.2.5.** An *$I_d$-determined directed acyclic graph from $q \in Q_1$* $DAG_{I_d}$ is a subgraph of $DAG_d$ from root $q$ in which the next $d$ moves of player $I$ are determined, i.e., each vertex for player $I$ on the same odd level in $DAG_{I_d}$ has only one outgoing edge with the same label.

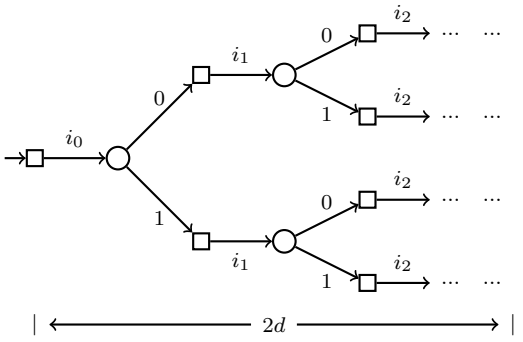


Figure 5.9: $I_d$-determined subgraph $T_{I_d}$ with $I_d = i_0 i_1 i_2 \dots i_{d-1} \in \{0,1\}^d$

Now we introduce an algorithm for solving the problem of whether player $O$ has a winning strategy with a bounded delay in a reachability games. Let $n = |Q_1|$ be the size of a game graph $G_1$. By the pigeonhole principle, the set of vertices on level $2^{n+1}$ is repeated on another level in each $DAG_{I_{2^n}}$ as shown in Figure 5.10 . Player $O$ has no winning strategy with a bounded delay in a reachability game, if and only if there is an $I_{2^n}$-determined directed acyclic graph $DAG_{I_{2^n}}$ containing no path that intersects with $F$. In such case player $I$ can therefore force all plays in $DAG_d$ from $q_j$ with an arbitrary shift $d > 0$ to avoid any vertex in $F$ by choosing the same sequence of input bits between two repeated levels in $DAG_{I_d}$. In other words, player $O$ has no winning strategy with a finite delay if and only if player $O$ can not force a play from $q_j$ to reach $F$ in each $DAG_{I_{2^n}}$.

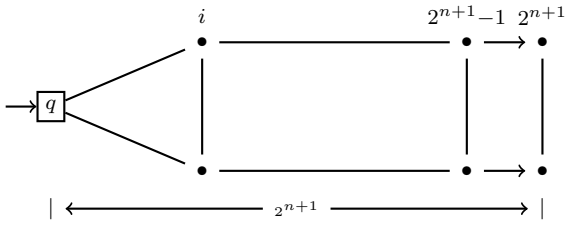


Figure 5.10: $I_{2^n}$-determined directed acyclic graph $DAG_{2^n}$ with root $q_j \in Q_1$

**Lemma 5.2.6.** *Player $I$ wins a reachability game with an arbitrary shift from vertex $q \in Q_1$ if and only if there is one $I_{2^n}$-determined directed acyclic*

*graph $DAG_{I_{2^n}}$ with root $q$ and $n = |Q_1|$ containing no path that intersects with $F$. In other words, player $O$ does not win a reachability game with a bounded delay if and only player $O$ has no winning strategy with delay $d$ for all $0 < d \leq 2^n$ in the reachability game.*

Player $O$ does not win a reachability game with an arbitrary delay if and only if player $O$ has no attractor winning strategy from all vertices in $I_d \times q_0$ in $G_d$ for all $d$ with $0 < d \leq 2^n$.

**Example 5.2.7.** We obtain a directed acyclic graph $DAG_3$ from the reachability game from 5.2.1, which is represented in Figure 4.3.

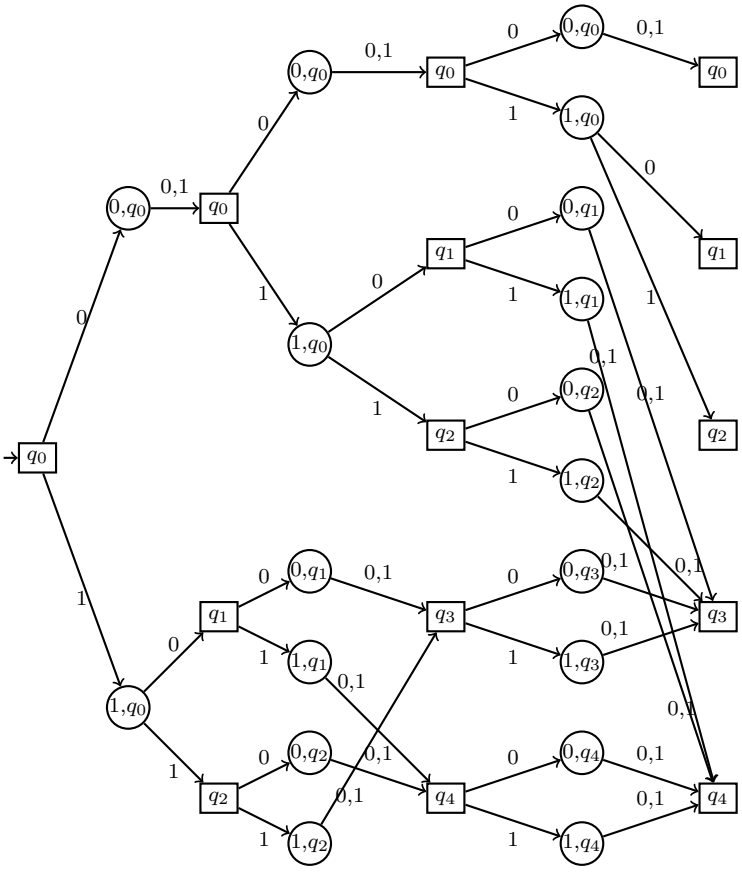


Figure 5.11: 3-delay directed acyclic graph $DAG_3$ with length 6

An $I_3$-determined directed acyclic graph of the same reachability game $DAG_{I_3}$ is shown in Figure 5.12. Since this $DAG_{I_3}$ consists of all paths not

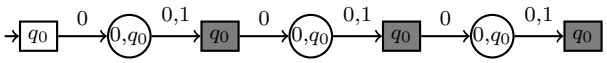


Figure 5.12: $I_3$-determined directed acyclic graph $DAG_{I_3}$ with length 6

intersecting with $F$, and the set of vertices is repeated on different levels for player $I$ in this $DAG_{I_3}$. Player $I$ can prevent reaching vertices in $F$ by repeating the same moves, i.e., player $I$ chooses 0 continuously as input bits infinitely often. It follows that player $O$ has no winning strategy with finite delay in this reachability game.

Obviously, there exist repeated levels in each $I_{32}$-determined directed acyclic graph $DAG_{I_{32}}$ of this reachability game.

## 5.3 Winning strategies with bounded delay in weak parity games

As for reachability games, in weak parity games the winning region $W_O$ of player $O$ increases monotonically with the increase of finite delay $d$, but not strictly monotonically either. Thus we need to consider the breaking condition for the construction of $d$-delay weak parity game graph $G_d$.

Player $O$ wins a weak parity game with delay $d$ from vertex $q \in Q_1$, if and only if on each $I_d$-determined directed acyclic graph $DAG_{I_d}$ with depth $2d$ there is at least one path from root $q$ on which the maximal color of the vertices is even. In other words, player $I$ has a winning strategy with delay $d$ in the weak parity game if and only if there is one $I_d$-determined directed acyclic graph $DAG_{I_d}$ with depth $2d$ containing a path from $q_j$ on which the maximal color of the vertices is odd. Player $O$ does not win a weak parity game with bounded delay if and only if player $O$ has no positional strategy to win from all vertices in $I_d \times q_0$ in $G_d$ for all $d$ with $0 < d \leq 2^n$, where $n = |Q_1|$ denotes the size of the weak parity game graph $G_1$. Hence, player $I$ wins a weak parity game $G$ with an arbitrary delay.

**Lemma 5.3.1.** *Player $I$ wins a weak parity game with an arbitrary shift from vertex $q \in Q_1$ if and only if there is one $I_{2^n}$-determined directed acyclic graph $DAG_{2^n}$ with root $q$ and $n = |Q_1|$ containing only paths consisting of vertices whose maximal color is odd. In other words, player $O$ does not win a*

*weak parity game with bounded delay if and only if player $O$ has no winning strategy with delay $d$ for all $d$ with $0 < d \leq 2^n$ in the weak parity game.*

## 5.4 Winning strategies with bounded delay in parity games

In order to introduce the algorithm which solves the problem whether a solution with a bounded delay in a parity game exists, we recall first the algorithmic solution of parity games in [17], which proves that parity games are positionally determined, as shown in Theorem 4.5.2. This proof is built up inductively over $n := |Q|$ for a general parity game graph $G = (Q, \mu)$ with a coloring function $c : Q \to \{0, 1, \ldots, k\}$:

*Proof.* The algorithm is defined inductively over $n := |Q|$.

1. $n = 2$: If the maximal color is even then $W_O = Q$, and if the maximal color is odd then $W_I = Q$, since the game graph is of the form:

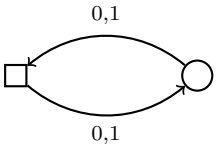


Figure 5.13: General parity game graph $G = (Q, \mu)$ with $|Q| = 2$

2. $n + 1$: We assume that the maximal color $k$ which occurs in the game graph is even. In the other case (the maximal color is odd) the induction step is analogous, with swapped roles of the two players. We pick a $q \in Q$ with the maximal even color in $G$. Notice that $Q \backslash \mathrm{Attr}_O^{|Q|}(\{q\})$ gives a game graph with less than $n$ vertices. The induction hypothesis provides a partition of $Q \backslash \mathrm{Attr}_O^{|Q|}(\{q\})$ into winning regions $U_I$ of player $I$ and $U_O$ of player $O$ respectively, and corresponding positional winning strategies for both players.

   - Case 1: Player $O$ can guarantee that a transition from $q$ to $U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\})$ is taken, which means:

(a) $q \in Q_O$ and $\exists q\ (q,p) \in E$ with $p \in U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\})$; or
(b) $q \in Q_I$ and $\forall q\ (q,p) \in E$ with $p \in U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\})$.

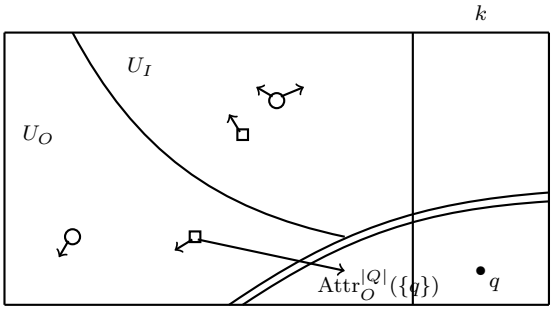


Figure 5.14: Parity game with $G = (Q, \mu)$ with $|Q| = n + 1$ in Case 1.

Hence we obtain:

(a) $U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\}) \subseteq W_O$: Because a play from $\mathrm{Attr}_O^{|Q|}(\{q\})$ either remains in $U_O$, i.e., player $O$ wins by induction hypothesis; or passes through $q$ infinitely often, i.e., player $O$ wins as the maximal color $c(q)$ in the play is even.
(b) $U_I \subseteq W_I$: Because player $I$ can guarantee that a play from vertices in $U_I$ remains in $U_I$. From vertices for player $I$ there is at least one edge leading to $U_I$, and from vertices for player $O$ all edges lead to $U_I$.

Because $U_O \cup U_I \cup \mathrm{Attr}_O^{|Q|}(\{q\}) = Q$, we have $W_O = U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\})$ and $W_I = U_I$.

The positional strategies for player $O$ on $U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\})$ and for player $I$ on $U_I$ are:

(a) Player $O$ plays on $U_O$ according to the positional strategy given by the induction hypothesis;
(b) Player $O$ plays on $\mathrm{Attr}_O^{|Q|}(\{q\})$ according to the attractor strategy;
(c) From $q$, player $O$ can guarantee a move back to $U_O \cup \mathrm{Attr}_O^{|Q|}(\{q\})$;
(d) Player $I$ has a positional strategy on $U_I$ given by the induction hypothesis.

- Case 2: Player $I$ can guarantee that a transition from $q$ to $U_I$ is taken, which means:
  (a) $q \in Q_O$ and $\forall q\ (q,p) \in E$ with $p \in U_I$; or
  (b) $q \in Q_I$ and $\exists q\ (q,p) \in E$ with $p \in U_I$.

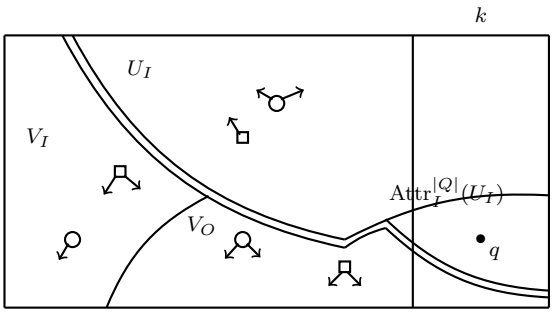


Figure 5.15: Parity game with $G = (Q, \mu)$ with $|Q| = n + 1$ in Case 2.

Hence, $q \in \mathrm{Attr}_I^{|Q|}(U_I)$. The induction hypothesis gives winning regions $V_I$ of player $I$ and $V_O$ of player $O$ respectively in the game graph $Q \backslash \mathrm{Attr}_I^{|Q|}(U_I)$ with less than $n$ vertices, and the corresponding positional strategies. We obtain:

(a) $V_O \subseteq W_O$ for the given positional winning strategy.

(b) $V_I \cup \mathrm{Attr}_I^{|Q|}(U_I) \subseteq W_I$: This holds because player $I$ can move from $p \in \mathrm{Attr}_I^{|Q|}(U_I)$ to $U_I$ and guarantees the play to remain in $U_I$. From $p \in V_I$ player $I$ can either move back to $V_I$ or move to $\mathrm{Attr}_I^{|Q|}(U_I)$. Thus, player $I$ wins in both cases.

Since $V_I \cup V_O \cup \mathrm{Attr}_I^{|Q|}(U_I) = Q$, we have $W_O = V_O$ and $W_I = V_I \cup \mathrm{Attr}_I^{|Q|}(U_I)$.

□

To show that the problem whether a parity games has a solution with a bounded delay, we follow the idea of this algorithm and proceed inductively over $n := |Q_1|$ for a parity game with the 1-delay game graph $G_1 = (Q_1, \mu_1)$ with a coloring function $c : Q_1 \to \{0, 1, \ldots, k\}$. We use $d$-delay game graphs $G_d$, which were introduced in the previous chapter, and $d$-delay absorbed game graphs $G^d$ defined in this chapter.

1. First we consider a parity game with the 1-delay game graph $G_1 = (Q_1, \mu_1)$ where $|Q_1| = 2$. The $d$-delay game graph $G_d$ contains $2^{d-1}$ subgraphs of the following form:

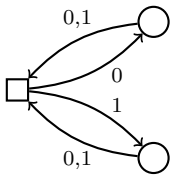


Figure 5.16: Subgraph of $G_d$ for a parity game with $G_1$ where $|Q_1| = 2$

If the maximal color in $G_d$ is even then $W_O^d = Q_d$, and if the maximal color in $G_d$ is odd then $W_I^d = Q_d$ in the $d$-delay absorbed game graph. For all $G_1 = (Q_1, \mu_1)$ with $|Q_1| = n$, we claim that for all $d$ with $d \geq 2^n$:

$$W_I^d = W_I^{2^n} \text{ and } W_O^d = W_O^{2^n}.$$

2. Now we construct winning strategies with finite delay for player $I$ and player $O$, in a parity game with $G_1 = (Q_1, \mu_1)$ with $|Q_1| = n+1$, and a coloring function $c : Q_1 \to \{0, 1, \ldots, k\}$. We assume that the maximal color $k$ which occurs in the parity game graph $G_1$ is even. In the other case (the maximal color is odd) the induction step is analogous, with swapped roles of the two players. We pick a $q \in Q_{I_1}$ with the maximal even color in $G_1$. Let $F_d$ be the set $I_{d-1} \times \{q\}$, the induction hypothesis provides a partition of $Q_d \backslash \text{Attr}_O^{|Q_d|}(F_d)$ into the winning regions $U_{I_d}$ of player $I$ and $U_{O_d}$ of player $O$ respectively, and the corresponding positional winning strategies with delay $d$.

   - Case 1: Player $O$ can guarantee that a transition from $q' \in F_d$ to $U_{O_d} \cup \text{Attr}_O^{|Q_d|}(F_d)$ is taken, which means:
     (a) $q' \in F_d \cap Q_{O_d}$ and $\exists q'\ (q', p') \in E_d$ with $p' \in U_{O_d} \cup \text{Attr}_O^{|Q_d|}(F_d)$; or
     (b) $q' \in F_d \cap Q_{I_d}$ and $\forall q'\ (q', p') \in E_d$ with $p' \in U_{O_d} \cup \text{Attr}_O^{|Q_d|}(F_d)$.

     Hence, we obtain in the $d$-delay absorbed game graph $G^d$ of the parity game:

(a) $U_{O_d} \cup \mathrm{Attr}_O^{|Q_d|}(F_d)$ defines $W_O^d$, because a play from $\mathrm{Attr}_O^{|Q_d|}(F_d)$ either remains in $U_{O_d}$, i.e., player $O$ will win by induction hypothesis; or passes through $F_d$ infinitely often, i.e., player $O$ wins as vertices in $F$ have the maximal even color.

(b) $U_{I_d}$ defines $W_I^d$, because player $I$ can guarantee that a play from vertices in $U_{I_d}$ remains in $U_{I_d}$. From the vertices of player $I$ there is at least on edge leading to $U_{I_d}$, and from the vertices for player $O$ all edges lead to $U_{I_d}$.

According to the definition of $d$-delay absorbed game graph we have $W_I^d \dot{\cup} W_O^d = Q_1$.

The positional strategies for player $O$ on $U_{O_d} \cup \mathrm{Attr}_O^{|Q_d|}(F_d)$ and for player $I$ on $U_{I_d}$ are:

(a) Player $O$ plays on $U_{O_d}$ according to the positional strategy given by the induction hypothesis;

(b) Player $O$ plays on $\mathrm{Attr}_O^{|Q_d|}(F_d)$ according to the attractor strategy;

(c) From $q$ player $O$ can guarantee a move back to $U_{O_d} \cup \mathrm{Attr}_O^{|Q_d|}(F_d)$;

(d) Player $I$ has the positional strategy on $U_{I_d}$ given by the induction hypothesis.

We construct the $d$-delay absorbed game graph $G'^d = (Q'_1, \mu'_1, l)$ from $Q_d \backslash \mathrm{Attr}_I^{|Q_d|}(F_d)$. Let $n' = |Q'_1|$, obviously $n' < n + 1$. For all $d$ where $d \geq 2^{n'}$, the induction hypothesis gives $W_I'^d = W_I'^{2^{n'}}$

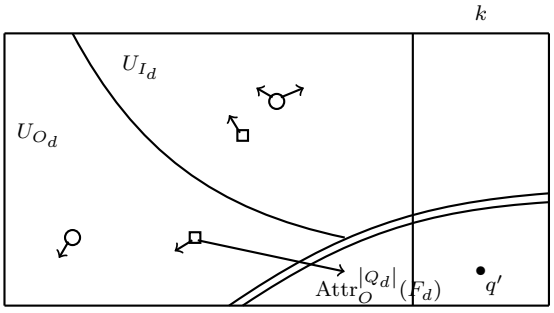


Figure 5.17: $d$-delay game graph $G_d$ of a parity game in Case 1.

and $W_O^{\prime d} = W_O^{2^{n'}}$ in the $d$-delay absorbed game graphs $G^{\prime d}$. Due to the result of reachability games, we have

$$\begin{aligned} \mathrm{Attr}_I^{|Q_{2^{n+1}}|}(F_{2^{n+1}}) &= \mathrm{Attr}_I^{|Q_{2^{n+1}+1}|}(F_{2^{n+1}+1}) = \ldots, \text{ and} \\ \mathrm{Attr}_O^{|Q_{2^{n+1}}|}(F_{2^{n+1}}) &= \mathrm{Attr}_O^{|Q_{2^{n+1}+1}|}(F_{2^{n+1}+1}) = \ldots \end{aligned}$$

where $|Q_1| = n + 1$. Hence, $W_I^d = W_I^{2^{n+1}}$ and $W_O^d = W_O^{2^{n+1}}$ hold for all $d$ where $d \geq 2^{n+1}$.

- Case 2: Player $I$ can guarantee that a transition from $q' \in F_d$ to $U_{I_d}$ is taken, which means:
  (a) $q' \in F_d \cap Q_{O_d}$ and $\forall q'\ (q', p') \in E_d$ with $p' \in U_{I_d}$; or
  (b) $q' \in F_d \cap Q_{I_d}$ and $\exists q'\ (q', p') \in E_d$ with $p' \in U_{I_d}$.

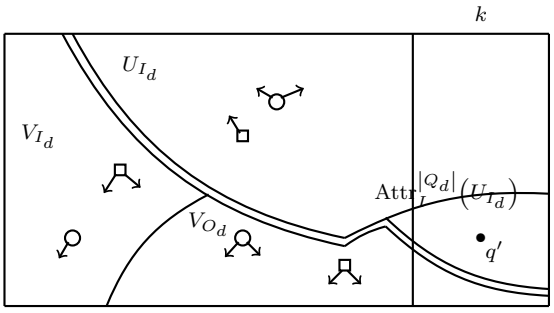


Figure 5.18: $d$-delay game graph $G_d$ of a parity game in Case 2.

Hence, $q' \in \mathrm{Attr}_I^{|Q_d|}(U_{I_d})$. The induction hypothesis gives the winning regions $V_{I_d}$ of player $I$ and $V_{O_d}$ of player $O$ respectively in a game graph $Q \backslash \mathrm{Attr}_I^{|Q_d|}(U_{I_d})$, and the corresponding positional strategies for both players. We obtain in the $d$-delay absorbed game graph $G^d$:

(a) $V_{O_d}$ defines $W_O^d$ for the given positional winning strategy.
(b) $V_{I_d} \cup \mathrm{Attr}_I^{|Q_d|}(U_{I_d})$ defines $W_I^d$, because player $I$ can move from $p \in \mathrm{Attr}_I^{|Q_d|}(U_{I_d})$ to $U_{I_d}$ and guarantees the play to remain

in $U_{I_d}$. From a vertex $p' \in V_{I_d}$ player $I$ can either move back to $V_{I_d}$ or move to $\mathrm{Attr}_I^{|Q_d|}(U_{I_d})$. Thus, player $I$ wins in both cases.

According to the definition of $d$-delay absorbed game graph we have $W_I^d \dot\cup W_O^d = Q_1$.
We the $d$-delay absorbed game graph $G'^d = (Q'_1, \mu'_1, l)$ from $Q_d \backslash \mathrm{Attr}_I^{|Q_d|}(U_{I_d})$. Let $n' = |Q'_1|$, obviously $n' < n+1$. For all $d$ where $d \geq 2^{n'}$, the induction hypothesis gives $W_I'^d = W_I'^{2^{n'}}$ and $W_O'^d = W_O'^{2^{n'}}$ in the $d$-delay absorbed game graphs $G'^d$. Due to the result of reachability games, we have

$$\begin{aligned} \mathrm{Attr}_I^{|Q_{2^{n+1}}|}\left(U_{I_{2^{n+1}}}\right) &= \mathrm{Attr}_I^{|Q_{2^{n+1}+1}|}\left(U_{I_{2^{n+1}+1}}\right) = \ldots, \text{ and} \\ \mathrm{Attr}_O^{|Q_{2^{n+1}}|}\left(U_{I_{2^{n+1}}}\right) &= \mathrm{Attr}_O^{|Q_{2^{n+1}+1}|}\left(U_{I_{2^{n+1}+1}}\right) = \ldots \end{aligned}$$

for $|Q_1| = n+1$. Hence, $W_I^d = W_I^{2^{n+1}}$ and $W_O^d = W_O^{2^{n+1}}$ hold for all $d$ where $d \geq 2^{n+1}$.

This algorithmic solution for parity games leads the following lemma.

**Lemma 5.4.1.** *Player $I$ wins a parity game with an arbitrary shift from vertex $q \in Q_1$ if and only if player $O$ has no winning strategy with delay $d$ for all $d$ with $0 < d \leq 2^n$ where $n = |Q_1|$. In other words, player $O$ does not win a parity game with bounded delay if and only player $O$ has no winning strategy with delay $d$ for all $d$ with $0 < d \leq 2^n$ in the parity game.*

Büchi games are a special case of parity games. For a Büchi game with a game graph $G = (Q, \mu)$, where $Q = Q_I \dot\cup Q_O$ and $F \subseteq Q$, the Büchi winning condition of player $O$ for a play $\rho$ is

$$\rho \in \mathrm{Win} :\Leftrightarrow \mathrm{Inf}(\rho) \cap F \neq \emptyset.$$

This game can be described as a special parity game over $G$ with coloring

$$c(q) = \begin{cases} 2, & \text{for} \quad q \in F \\ 1, & \text{for} \quad q \notin F \end{cases}.$$

This implies the following lemma for Büchi games.

**Lemma 5.4.2.** *Player I wins a Büchi game with an arbitrary shift from vertex $q \in Q_1$ if and only if player O has no winning strategy with delay $d$ for all $d$ with $0 < d \leq 2^n$ where $n = |Q_1|$. In other words, player O does not win a Büchi game with bounded delay if and only player O has no winning strategy with delay $d$ for all $d$ with $0 < d \leq 2^n$ in the Büchi game.*

**Theorem 5.4.3.** *Given a game graph $G = (Q, \mu)$ of a parity game with a coloring function $c : Q \to \{0, 1, \ldots, k\}$, the problem whether player O wins the parity game from a vertex $q \in Q$ with bounded delay is decidable.*

*Proof.* We have shown in this section that a consideration of a solution with delay $2^n$ is sufficient to determine whether player $O$ has a solution with bounded delay in an infinite game, where $n$ denotes the size of the corresponding game graph. □

As a consequence, we obtain a new proof for the result of Hosch and Landweber that the existence of the solution of Church's Problem by a strategy with a bounded delay is decidable.

## 5.5 Alternative approaches for finding solutions with bounded delay

We sketch an alternative approach without further details, towards a solution of the problem whether a solution with a bounded delay in a Büchi game exists. We consider $\omega$-regular languages in the form of $U \cdot V^\omega$ for some regular languages $U, V \in \Sigma^*$, which are Büchi recognizable.

**Definition 5.5.1.** Let $G_1 = (Q_1, \mu_1)$ be a 1-delay graph for a Büchi game. We define the *syntactic equivalence* relation in Büchi games over infinite words, denoted by $\sim$: $u \sim v$ if and only if, for all $p, q \in Q_1$

- $p \xrightarrow{u} q \;\Leftrightarrow\; p \xrightarrow{v} q$ and
- $p \xrightarrow{u}_F q \;\Leftrightarrow\; p \xrightarrow{v}_F q$,

where $p \xrightarrow{u} q$ denotes a run of $G_1$ from $p$ via $u$ to $q$, and $p \xrightarrow{u}_F q$ denotes a run of $G_1$ from $p$ via $u$ to $q$ that passes through some states in $F$.

Finite-state game graphs of infinite games contain only a finite number of such equivalence classes. Let $w_1, w_2, \ldots, w_k$ be some representatives of the equivalence classes. Ramsey type Lemma shows a certain periodicity of each $\omega$-word with respect to a finite partition of $\Sigma^*$.

**Lemma 5.5.2** ([16]). *Each $\omega$-regular language $\alpha \in \Sigma^\omega$ is decomposable as $\alpha = v_0 v_1 v_2 \ldots$ such that all $v_i$ for $i > 0$ are in the same equivalence class denoted by $V$.*

We restrict ourselves to sets $V$ with the property $V \cdot V \subseteq V$.

**Definition 5.5.3.** We call tuples of equivalence classes of the equivalence relation $\sim$ over finite words *building blocks*. A building block $(U, V)$ is *bad*, if for representatives $u \in U$ and $v \in V$, player $O$ can not find output bits to fulfill the Büchi condition according to the input sequence $uvvv \ldots \in \Sigma^\omega$.

**Definition 5.5.4.** A *$w$-determined directed acyclic graph $DAG(w)$* where $w \in \Sigma_I^*$ denotes a finite word, is a subgraph of $|w|$-delay directed acyclic graph $DAG_{|w|}$, in which the input sequence is determined by $w$.

Player $I$ wins the Büchi game with an arbitrary shift if and only if there exists a bad tuple $(U, V)$. Player $O$ wins the Büchi game if and only if for all building blocks $(U, V)$ of finitely many equivalence classes where $u$ is a representative of equivalence class $U$, and $v$ is a representative of equivalence class $V$, each $u \cdot v$-determined directed acyclic graph $DAG(u \cdot v)$ contains at least one path intersecting with $F$ between level $2|v| + 1$ and level $2|u \cdot v|$.

**Lemma 5.5.5.** *Let $l$ be the maximal length of the representations of the equivalence classes in a Büchi game $w_1, w_2, \ldots, w_k$, i.e., $l = \max\{|w_1|, |w_2|, \ldots, |w_k|\}$. Player $I$ wins the Büchi game with an arbitrary shift if and only if there is an $I_{2l}$-determined directed acyclic graph $DAG_{I_{2l}}$ with length $4l$ containing no path reaching $F$ between level $2l + 1$ and level $4l$. In other words, player $O$ does not win a Büchi game with bounded delay if and only if player $O$ has no winning strategy with delay $d$ for $0 < d \leq 2l$ in the Büchi game.*

Röll [14] provides an algorithm for determining the maximal length of representatives of equivalence classes in Büchi games.

We also sketch an alternative approach without further details, towards a solution of the problem whether a solution with a bounded delay in a parity

game exists. At first we define syntactic equivalence relation in parity games refer to [2].

**Definition 5.5.6.** Let $G_1 = (Q_1, \mu_1)$ be a 1-delay graph for a parity game with a coloring function $c \to \{0, 1, \ldots, k\}$ where $k$ is even. We define the *syntactic equivalence* relation in parity games over infinite words, denoted by $\sim_i$: $u \sim_i v$ if and only if, for all $p, q \in Q_1$

$$p \xrightarrow{u}_i q \;\Leftrightarrow\; p \xrightarrow{v}_i q;$$

where $p \xrightarrow{u}_i q$ denotes a run of $G_1$ from $p$ via $u$ to $q$ containing the maximal color $i \in \{0, 1, \ldots, k\}$ on the path over $u$ in the corresponding game graph, i.e., $i = \max\{c(\rho(u))\}$.

As we previously mentioned, each $\omega$-regular language $\alpha \in \Sigma^\omega$ is decomposable as $\alpha = v_0 v_1 v_2 \ldots$. such that all $v_i$ for $i > 0$ are in the same equivalence class denoted by V. We restrict ourselves to sets $V$ with the property $V \cdot V \subseteq V$.

**Lemma 5.5.7.** *Let $l$ be the maximal length of the representatives of the equivalence classes in this parity game $w_1, w_2, \ldots, w_k$, i.e., $l = \max\{|w_1|, |w_2|, \ldots, |w_k|\}$. Player I wins the parity game with an arbitrary shift if and only if there is an $I_{2l}$-determined directed acyclic graph $\mathrm{DAG}_{I_{2l}}$ with length $4l$ containing the maximal odd color on the path odd between level $2l+1$ and level $4l$. In other words, player O does not win a parity game with bounded delay if and only if player O has no winning strategy with delay $d$ for $0 < d \leq 2l$ in the parity game.*

# Chapter 6

# Conclusion

We study Church's Problem in the framework of infinite games in this thesis. Church's Problem was raised in [5] by Church and is considered as a master problem in computer science. Our main purpose is to determine solutions with delay for conditions in sequential calculus, i.e., to solve the problem of deciding whether Church's Problem with conditions stated in sequential calculus is solvable with delay. Even and Meyer solved the finite delay problem for sequential Boolean equations, which provides a solution to determine finite-delay solutions in safety games. Based on their work, Hosch and Landweber [8] solved the problem of the existence of solutions with finite delay for Church's Problem. In this thesis, we concentrate furthermore on finding finite delay solutions in arbitrary infinite games:

1. Whether a winning strategy with a fixed delay for player $O$ in an infinite game exists, and how to construct the corresponding winning strategy and winning region of player $O$?

2. How to solve the problem of whether player $O$ has winning strategy with a finite delay in an infinite game; in other words, does player $I$ win the infinite game with a finite shift?

To solve the first question we introduce $d$-delay game graphs $G_d$, which are constructed from 1-delay game graphs $G_1 = (Q_1, \mu_1)$ (i.e., general game graphs in this thesis). In $d$-delay reachability game graphs $G_d$, we can construct attractor winning strategies for player $O$ in reachability games, we call them $d$-delay attractor strategies. Therefore corresponding winning regions can be determined in $G_d$. To determine solutions with a fixed delay in

Büchi games or parity games, we construct positional winning strategies for player $O$ in $d$-delay Büchi game graph $G_d$ and $d$-delay parity game graph $G_d$ respectively.

For the second question, we present algorithms to determine the existence of solutions with bounded delay for player $O$. In arbitrary infinite regular games, player $O$ has winning strategies with bounded delay in an infinite game, if and only if player $O$ has a $d$-delay winning strategy for some $d$ with $0 < d \leq 2^{|Q_1|}$. Player $I$ has a winning strategy with an arbitrary shift (even an infinite shift) in a infinite game, if and only if player $I$ has $d$-shift winning strategies for all $d$ with $0 < d \leq 2^{|Q_1|}$. As a consequence, we obtain a new proof for the result of Hosch and Landweber that the existence of the solution of Church's Problem by a strategy with q bounded delay is decidable.

# Bibliography


[1] J. R. Büchi, C. C. Elgot: *Decision Problems of Weak Second-Order Arithmetics and Finite Automata*, Abstract 553-112, Notices Amer. Math. Soc. 5 (1958), 834.

[2] V. Bárány, L. Kaiser, S. Rubin: *Cardinality and Counting Quantifiers on Omega-Automatic Structures*, Im Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science, STACS 2008 (S. Albers and P. Weil, Eds.), pp. 385-396, 2008.

[3] J. R. Büchi, L. H. Landweber: *Solving Sequential Conditions by Finite State Strategies*, Trans. Amer. Math. Soc. 138(1969), 295-311.

[4] J. R. Büchi: *On a Decision Method in Restricted Second Order Arithmetic*, Proc. 1960 International Congress on Logic, Methodology and Philosophy of Science, E. Nagel at al., eds, Stanford University Press 1962, pp. 1-11.

[5] A. Church: *Applications of Recursive Arithmetic to the Problem of Circuit Synthesis*, Summaries of the Summer Institute of Symbolic Logik, Cornell Univ., Irhaca, N.Y. 1957, Volume I, pp. 3-50.

[6] S. Even, A. R. Meyer: *Sequential Boolean Equations*, IEEE Trans. Computers, C-18, No. 3 (1969), 230-240.

[7] D. Gale, F. M. Stewart: *Infinite Games with Perfect Information*, Ann. Math. Studies 28, 245-266 (1953).

[8] F. A. Hosch, L. H. Landweber: *Finite Delay Solutions for Sequential Conditions*, In: Nivat, M. (ed.) Proc. ICALP 1972, pp. 45-60. North-Holland, Amsterdam(1972).

[9] J. E. Hopcroft, R. Motwani, J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation*, 2nd or above editions, Addison-Wesley, 2001.

[10] R. McNaughton: *Finite-State Infinite Games*, Project MAC Rep., MIT, Cambridge, Mass., Sept. 1965.

[11] R. McNaughton: *Testing and Generating Infinite Sequences by a Finite Automaton*, Inf. Contr. 9(1966), 521-530.

[12] G. H. Mealy: *A Method for Synthesizing Sequential Circuits*, Bell Systems Technical Journal 34 (September 1955), 1045–1079.

[13] D. E. Muller: *Infinite Sequences and Finite Machines*, Proc. 4th IEEE Ann. Symp. on Switching Circuit Theory and Logical Design, IEEE Press 1963, pp. 3-16.

[14] A. Röll: *Complementation of Büchi Automata - Algorithms and Implementation*, Diplomarbeit, Logik und Theorie diskreter Systeme, RWTH-Aachen, 2008.

[15] B. A. Trakhtenbrot, Ya. M. Barzdin: *Finite Automata Behavior and Synthesis*, Fundamental Studies in Computer Science, 1973.

[16] Wolfgang Thomas: *Automata and Reactive Systems*, RWTH Aachen, Lehrstuhl Informatik 7, 2003, 70-72.

[17] Wolfgang Thomas: *Church's Problem and a Tour through Automata Theory*, RWTH Aachen, Lehrstuhl Informatik 7, 2007.

[18] B. A. Trakhtenbrot: *On Operators Realizable in Logical Nets*, Dokl. Akad. Naut. SSSR 112(1957), 1005-1007 [in Russian].

[19] B. A. Trakhtenbrot: *Synthesis of Logical Nets Whose Operators are Described of Monadic Predicates*, Dokl. Akad. Naut. SSSR 118(1958), 646-649 [in Russian].

# Index