# Degrees of Lookahead in Regular Infinite Games

Michael Holtmann[1], Łukasz Kaiser[2], and Wolfgang Thomas[1]

[1] RWTH Aachen, Lehrstuhl für Informatik 7, D-52056 Aachen
{holtmann,thomas}@automata.rwth-aachen.de
[2] RWTH Aachen, Mathematische Grundlagen der Informatik, D-52056 Aachen
kaiser@logic.rwth-aachen.de

**Abstract.** We study variants of regular infinite games where the strict alternation of moves between the two players is subject to modifications. The second player may postpone a move for a finite number of steps, or, in other words, exploit in his strategy some lookahead on the moves of the opponent. This captures situations in distributed systems, e.g. when buffers are present in communication or when signal transmission between components is deferred. We distinguish strategies with different degrees of lookahead, among them being the continuous and the bounded lookahead strategies. In the first case the lookahead is of finite possibly unbounded size, whereas in the second case it is of bounded size. We show that for regular infinite games the solvability by continuous strategies is decidable, and that a continuous strategy can always be reduced to one of bounded lookahead. Moreover, this lookahead is at most doubly exponential in the size of the parity automaton recognizing the winning condition. We also show that the result fails for non-regular games where the winning condition is given by a context-free $\omega$-language.

## 1 Introduction

The algorithmic theory of infinite games is a powerful and flexible framework for the design of reactive systems (see e.g. [1]). It is well known that, for instance, the construction of a controller acting indefinitely within its environment amounts to the computation of a winning strategy in an infinite game. For the case of regular games, algorithmic solutions of this synthesis problem have been developed, providing methods for automatic construction of controllers. The basis of this approach is the Büchi-Landweber Theorem, which says that in a regular infinite game, i.e. a game over a finite arena with a winning condition given by an $\omega$-regular language, a finite-state winning strategy for the winner can be constructed [2]. Much work in the past two decades has been devoted to generalizations of this fundamental result. The game-theoretic setting is built on two components, a *game arena* or game graph, representing the transition structure of a system, and a *winning condition*, usually given by a logic formula or an automata theoretic condition. Most generalizations address an extension of either of the two, or both. A rapidly growing literature is thus concerned with the case of infinite game graphs and non-regular winning conditions [3, 4, 5].

In the present paper we investigate a different kind of generalization of the basic setting, regarding the possibility to get a lookahead on the moves of the opponent. To explain this aspect it is convenient to refer to the simplest format of infinite games, also called Gale-Stewart games [6]. In such a game we abstract from arenas but just let the two players choose letters from a finite alphabet in turn. (For notational convenience let us only consider the typical case of the Boolean alphabet $\mathbb{B} := \{0, 1\}$.) A play is built up as a sequence $a_0 b_0 a_1 b_1 \cdots$ where $a_i$ is chosen by one player and $b_i$ by the other. A natural view is to consider the sequence $\alpha = a_0 a_1 \cdots$ as *input stream* and $\beta = b_0 b_1 \cdots$ as *output stream*. Accordingly, the players are called Player Input and Player Output, or short Player I and Player O. The play is won by Player O if the $\omega$-word $\alpha^\wedge\beta := \binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots \in (\mathbb{B}^2)^\omega$ satisfies the winning condition, i.e. if it belongs to a given $\omega$-regular language $L$. In the classical setting, a strategy for Player O is a function $f$ that maps a finite input prefix $a_0 \cdots a_i$ to the bit $b_i$ that is to be chosen by Player O. Such a strategy induces an operator $\lambda : \mathbb{B}^\omega \to \mathbb{B}^\omega$ from input streams to output streams. In this work we study more generalized operators that correspond to strategies where the choice of $b_i$ depends on $a_0 \cdots a_j$, for $j \neq i$. We show results on the existence of such strategies for different conditions on the relation between $i$ and $j$.

There are two motivations for the study of such a generalization, a practical and a theoretical one. In many scenarios, the occurrence of delays (say between input and output) is realistic, either as a modeling assumption or as a feature of strategies. For example, the design of a controller may involve a buffer that allows to store a sequence of input bits of some fixed length $d$ such that the bit $b_i$ of the output sequence is to be delivered with lookahead $d$, i.e. on the basis of the input sequence $a_0 \cdots a_{i+d}$. Conversely, in the context of networked control (i.e. systems with components in different locations), there may be a delay $d$ in the transmission of data, which means that the delivery of $b_i$ is due at a point where only the input bits $a_0 \cdots a_{i-d}$ are available. It is clear that the occurrence of lookaheads and delays influences the existence of solutions. In the first case, we obtain for increasing $d$ an increasing advantage for the output player, whereas in the second case we obtain an increasing disadvantage. Observe that the cases are symmetric and thus mutually reducible.

A more theoretical motivation is to explore more comprehensively and systematically the solution concepts for infinite games. The classical concept of a strategy gives a very special kind of operator, and there are very natural options of higher generality, well-known already from background fields like descriptive set theory and topology [6]. Let us mention four fundamental levels of operators, corresponding to different levels of obligation for Player O to move. The most general ones are the continuous operators (see e.g. [7, 8]). An operator $\lambda$ is continuous (in the Cantor space of infinite sequences over $\mathbb{B}$) if in the output sequence $\beta = \lambda(\alpha)$ the bit $b_i$ is determined by a finite prefix of $\alpha$. Referring only to the length of prefixes, we call an operator uniformly continuous if for some function $g : \mathbb{N} \to \mathbb{N}$ we have that $b_i$ is determined by $a_0 \cdots a_{i+g(i)}$. For fixed $g$ we then speak of $g$-delay operators. On a further level of specialization, we

are dealing with operators of bounded delay; these are $g$-delay operators for a constant function $g$. Finally, the constant 0 supplies the operators induced by standard strategies. The aforementioned levels naturally correspond to different types of games; for example, a continuous strategy involves the values "wait" and "output $b$" after each move of the opponent.

Our main result connects the aforementioned levels of operators in the context of regular games. We show that in a Gale-Stewart game with regular winning condition, one can decide whether there is a continuous winning strategy for Player O, and in this case a strategy of bounded delay can be constructed. Moreover, one can compute a suitable bound $d$ for the delay. Thus, in the first aforementioned application scenario, if a standard controller for satisfying a regular specification does not exist then one can decide whether some finite buffer will help, and determine the needed size of that buffer. We also show that the result fails when passing to non-regular specifications. However, which functions may be appropriate for uniformly continuous strategies in the non-regular case is left open. It seems that for infinite-state (or non-regular) games our result can serve as an entry into a much wider field of study.

As indicated above, the idea of generalized concepts of strategies is far from new. An early contribution is found in the (not well-known) paper of Hosch and Landweber [9]. It deals with bounded delay strategies in regular games and exploits a result of Even and Meyer from Boolean circuit theory to establish a bound for delays [10]. We obtain this result here as a corollary of the main theorem. The extension of our result over [9] covers three dimensions: the connection with strategies of unbounded delay, a considerably simplified and transparent proof of the Hosch-Landweber-Theorem (the construction in [9] is highly complex), and finally better complexity bounds for suitable delays.

This paper is organized as follows. In the next section we introduce notation. In Section 3 we present several kinds of functions and the operators they induce. We also bridge from continuous operators to delay operators and introduce games with delay. In Sections 4–6 we prove our main result via a twostage reduction: In Section 4 we do the first step, switching over to block games. In Section 5 we deal with notions related to semigroups and define a semigroup game. This framework is finally used in Section 6 to establish the second step of the reduction, i.e. the connection between block games and the semigroup game. Sections 7 and 8 provide evidence that our results cannot be generalized to $\omega$-context-free specifications and give an outlook on future investigations.

## 2 Preliminaries

Let $\Sigma$ be a finite *alphabet*. By $\Sigma^*$ and $\Sigma^\omega$ we denote the sets of finite and infinite *words* over $\Sigma$. Usually, finite words are denoted $u, v, \ldots$ whereas $\alpha, \beta, \ldots$ are infinite words. By $|u|$ we denote the *length* of $u$ and $\Sigma^n := \{u \mid |u| = n\}$ is the set of words of length $n$. $\mathbb{N}$ is the set of natural numbers and $\mathbb{N}_+ := \mathbb{N} \setminus \{0\}$. Given $n_1, n_2 \in \mathbb{N}$ with $n_1 < n_2$ we write $\Sigma^{[n_1,n_2]}$ for $\bigcup_{n_1 \leq n \leq n_2} \Sigma^n$.

A *(deterministic) finite automaton*, DFA for short, over $\Sigma$ is a tuple $\mathcal{A} = (Q, q_0, \delta, F)$ where $Q$ is a (non-empty) finite set of *states*, $q_0 \in Q$ is the *initial state*, $\delta : Q \times \Sigma \to Q$ is a *transition function*, and $F \subseteq Q$ is a set of *final states*. The *run* $\rho_u$ of $\mathcal{A}$ on $u := u_0 \cdots u_{n-1}$ is the finite sequence $\rho_u(0) \cdots \rho_u(n)$ with $\rho_u(0) = q_0$ and $\rho_u(i+1) = \delta(\rho_u(i), u_i)$ for $i = 0, \ldots, n-1$. We define $\mathcal{A}$ to *accept* $u$ if and only if $\rho_u(n) \in F$. The set of all words accepted by $\mathcal{A}$ is called the *$*$-language* of $\mathcal{A}$ and denoted $L_*(\mathcal{A})$. Later in our work we need the following basic property of deterministic finite automata.

**Lemma 1.** *Let $\mathcal{A}$ be a DFA with $n$ states and $|L_*(\mathcal{A})| = \infty$. Then, for all $i \in \mathbb{N}$, $\mathcal{A}$ accepts a word $u_i$ of length $i \leq |u_i| \leq i + n$.*

A *(deterministic) parity automaton*, DPA for short, over $\Sigma$ is similar to a DFA, but instead of the set $F$ of final states it has a *coloring*, i.e. a function $c : Q \to \{0, \ldots, m\}$. A run of a DPA is the natural extension of a run of a DFA to infinite words. For $\alpha \in \Sigma^\omega$, the set $\mathrm{Inf}(\rho_\alpha)$ is the set of states visited infinitely often in run $\rho_\alpha$. We define the parity automaton $\mathcal{A}$ to accept $\alpha$ if and only if $\max(\mathrm{Inf}(c(\rho_\alpha)))$ is even, i.e. the maximal color seen infinitely often in the run on $\alpha$ is even. Accordingly, the acceptance condition of $\mathcal{A}$ is called a *max-parity* acceptance condition. The set of all words accepted by $\mathcal{A}$ is called the $\omega$-language of $\mathcal{A}$ and denoted $L_\omega(\mathcal{A})$.

In the sequel, we write $L(\mathcal{A})$ instead of $L_*(\mathcal{A})$ or $L_\omega(\mathcal{A})$, if it is clear from the context whether $\mathcal{A}$ is a DFA or DPA. It is well-known that the class of languages accepted by DPA is exactly the class of *$\omega$-regular* languages (see e.g. [1]).

A *parity game* $\Gamma = (V, V_I, V_O, E, c)$ is played by two players, Player I and Player O, on a directed graph $G = (V, E)$:

- $V = V_I \uplus V_O$ is a partition of $V$ into positions of Player I and Player O,
- $E \subseteq V \times V$ is the set of allowed *moves*, and
- $c : V \to \{0, \ldots, m\}$ is a coloring of $V$ (w.l.o.g. $m \in 2\mathbb{N}$).

We assume that for each $v \in V$ there is a valid move from $v$, i.e. $vE := \{w \mid (v, w) \in E\} \neq \emptyset$. A *play* is an infinite path through $G$. A (standard) *strategy* for Player O is a function $f : V^* V_O \to V$ defining, for each position of Player O and each history $v_0 \cdots v_k$ of the play, her next move. Thus, for each $v_0 \cdots v_k$ (with $(v_i, v_{i+1}) \in E$ for all $i = 0, \ldots, k-1$) and $v_k \in V_O$, the function $f$ is defined such that $(v_k, f(v_0 \cdots v_k)) \in E$. A play $v_0 v_1 \cdots$ is *consistent* with the strategy $f$ if for each $v_i \in V_O$ the next position is given by $f$, i.e. $v_{i+1} = f(v_0 \cdots v_i)$.

The *parity winning condition* is again defined so that a play $v_0 v_1 \cdots$ is winning for Player O if and only if the maximal color occurring infinitely often in $\{c(v_i) \mid i \in \mathbb{N}\}$ is even. In the other case the play is winning for Player I. The function $f$ is called a *winning strategy for Player O from $v_0$* if each play starting in $v_0$ that is consistent with $f$ is winning for Player O, and analogously for Player I. Parity games, even on infinite graphs, are *determined*, i.e. for each $v$ either Player I or Player O has a winning strategy from $v$ (see e.g. [1]).

For the rest of this paper, let us fix $\{0, 1\}$ as input and output alphabet, i.e. let $\Sigma_I = \Sigma_O := \mathbb{B}$. All the definitions and results are analogous for other finite alphabets of size at least two.

## 3 Operators and Games with Delay

In this section we introduce different kinds of functions and operators, and show how they induce games with different degrees of lookahead. In the sequel, we mostly use the term "delay" in place of "lookahead", following e.g. [9].

### 3.1 Delay Operators

Let $\lambda$ denote a function from $\mathbb{B}^\omega$ to $\mathbb{B}^\omega$, also called an *operator*. We distinguish the following classes of operators, starting form the most general ones.

(1) *continuous operators*
(2) *uniformly continuous operators*
(3) *$f$-delay operators* for a fixed $f : \mathbb{N} \to \mathbb{N}$
(4) *bounded delay operators*
(5) *$d$-delay operators* for a fixed $d \in \mathbb{N}$

An operator $\lambda$ is continuous if in the output sequence $\beta = \lambda(\alpha)$ each bit is determined by a finite prefix of $\alpha$. This definition is equivalent to the topological one, i.e. $\lambda$ is continuous if the preimage $\lambda^{-1}(U)$ of every open set $U \subseteq \mathbb{B}^\omega$ is open in $\mathbb{B}^\omega$. Open sets in $\mathbb{B}^\omega$ are defined using the standard Cantor topology, i.e. $U \subseteq \mathbb{B}^\omega$ is open if there exists $W \subseteq \mathbb{B}^*$ such that $U = \{w\mathbb{B}^\omega \mid w \in W\}$. To formally capture the constraint that each output bit is determined by a finite prefix of the input, we define continuity using labelings of the full binary tree with $\{0, 1\}$ and $\bot$, where $\bot$ should be understood as "wait for next bit".

**Definition 1.** *An operator $\lambda : \mathbb{B}^\omega \to \mathbb{B}^\omega$ is* continuous *if there exists $l : \mathbb{B}^* \to \{0, 1, \bot\}$ such that for all $\alpha \in \mathbb{B}^\omega$ the word $l(\alpha) := l(\alpha_0)l(\alpha_0\alpha_1)l(\alpha_0\alpha_1\alpha_2)\cdots$ satisfies the following:*

*(1) $l(\alpha)$ does not end with $\bot^\omega$, and*
*(2) $\lambda(\alpha) = strip(l(\alpha))$ where $strip(l(\alpha))$ is the word $l(\alpha)$ with all $\bot$ removed.*

Let us now define *$f$-delay* and *uniformly continuous* operators. Let $f : \mathbb{N} \to \mathbb{N}$ be a strictly monotone function. We say that $\lambda$ is an *$f$-delay* operator if, for each $\alpha \in \mathbb{B}^\omega$, the bit $\lambda(\alpha)_i$ depends only on $\alpha_0 \cdots \alpha_{f(i)}$. An operator $\lambda$ is uniformly continuous if there exists an $f$ such that $\lambda$ is an $f$-delay operator.

Observe that each uniformly continuous operator is indeed continuous. To see this, simply label the first $f(0) - 1$ levels of the binary tree with $\bot$ and then each node $\alpha_0 \cdots \alpha_{f(0)}$ on level $f(0)$ with the appropriate first bit of $\lambda(\alpha_0 \cdots \alpha_{f(0)}\beta)$. Note that, for any $\beta$, this bit only depends on $\alpha_0 \cdots \alpha_{f(0)}$. Then label levels $f(0) + 1$ to $f(1) - 1$ with $\bot$ and level $f(1)$ with the second bits, and so on.

For the space $\mathbb{B}^\omega$ it is known that the converse also holds. This is a consequence of König's Lemma, or equivalently of the fact that continuous functions on a closed bounded space are uniformly continuous.

**Lemma 2.** *For every continuous operator $\lambda : \mathbb{B}^\omega \to \mathbb{B}^\omega$ there exists a strictly monotone function $f : \mathbb{N} \to \mathbb{N}$ such that $\lambda$ is an $f$-delay operator.*

By the above lemma, the classes of continuous operators $\mathbb{B}^\omega \to \mathbb{B}^\omega$ and uniformly continuous operators $\mathbb{B}^\omega \to \mathbb{B}^\omega$ are exactly the same. We distinguish them because our results rely in fact on uniform continuity. For example, consider the following operator $\lambda_1 : \mathbb{B}^\omega \setminus \{0^\omega\} \to \mathbb{B}^\omega$:

$$\lambda_1(\alpha) := \begin{cases} 0^\omega & \text{if } \alpha = 0^*10\beta \text{ for some } \beta \in \mathbb{B}^\omega \\ 1^\omega & \text{otherwise} \end{cases}$$

Intuitively, the operator $\lambda_1$ checks if there is a 0 or a 1 after the first 1 in the input. One can verify that $\lambda_1$ is a continuous function from $\mathbb{B}^\omega \setminus \{0^\omega\}$ to $\mathbb{B}^\omega$, but it is not uniformly continuous and can not be extended to any continuous function from $\mathbb{B}^\omega$ to $\mathbb{B}^\omega$. Our results do not hold for such operators: Already $\lambda_1$ is a counterexample, since it is continuous but not of bounded delay.

Among the uniformly continuous operators, we distinguish an even more restricted class of bounded delay operators. A function $f : \mathbb{N} \to \mathbb{N}$ is said to be of *bounded delay* if there exist $i_0, d \in \mathbb{N}$ such that $f(i) = i + d$ for all $i \geq i_0$, and it is said to be a *d-delay* function (or a function of *constant* delay $d$) if $f(i) = i + d$ for all $i \in \mathbb{N}$. The induced operators are named accordingly.

In all definitions above, we assume that the delay functions $f : \mathbb{N} \to \mathbb{N}$ are strictly monotone. For such a function $f$ it is often comfortable to consider $f^\Delta$, denoting the number of bits Player I has to choose in each round:

$$f^\Delta(i) := \begin{cases} f(0) + 1 & \text{if } i = 0 \\ f(i) - f(i-1) & \text{if } i > 0 \end{cases}$$

For strictly monotone $f : \mathbb{N} \to \mathbb{N}$ we obtain a unique $f^\Delta : \mathbb{N} \to \mathbb{N}_+$, and vice versa. For technical convenience, from now on we work only with the functions $f^\Delta : \mathbb{N} \to \mathbb{N}_+$ (but we omit the superscript $\Delta$ in our notation). Moreover, we use a special notation for constant delay functions: $\langle d \rangle$ denotes the function $g$ such that $g(0) = d + 1$ and $g(i) = 1$ for $i > 0$.

### 3.2 Regular Games with Delay

In this section we introduce the regular infinite game $\Gamma_f(L)$ (or $\Gamma_f(\mathcal{A})$). It is induced by an $\omega$-language $L$ (or a DPA $\mathcal{A}$) over $\mathbb{B}^2$ and a function $f : \mathbb{N} \to \mathbb{N}_+$. (Since we focus on the impact of the function $f$, we omit $L$ (or $\mathcal{A}$) if it is clear from the context and write $\Gamma_f$.) The function $f$ imposes a delay (or lookahead) on the moves of Player O. This means that in round $i$ Player I has to choose $f(i)$ many bits, and Player O chooses one bit, afterwards. This way the players build up two infinite sequences; Player I builds up $\alpha$ and Player O builds up $\beta$, respectively. The corresponding play is winning for Player O if and only if the word $\alpha^\wedge\beta := \binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots$ is accepted by $\mathcal{A}$. For a DPA $\mathcal{A}$, we say that $L(\mathcal{A})$ is *solvable with finite delay* if and only if there exists $f : \mathbb{N} \to \mathbb{N}_+$ such that Player O wins $\Gamma_f(L(\mathcal{A}))$ (analogously for restricted classes of functions).

Observe that the possible strategies for Player O in $\Gamma_f$ correspond precisely to $f$-delay operators, since Player O must output her $i$th bit after receiving

the next $f(i)$ bits of input. Thus, the question whether there exists an $f$-delay operator $\lambda$ such that $\{\binom{\alpha}{\lambda(\alpha)} \mid \alpha \in \mathbb{B}^\omega\} \subseteq L(\mathcal{A})$ is equivalent to the question whether there exists a winning strategy for Player O in $\Gamma_f$.

A basic observation is that winning with delay is a monotone property. For two functions $f, g : \mathbb{N} \to \mathbb{N}_+$ we write $f \sqsubseteq g$ if and only if $f(i) \leq g(i)$ for all $i \in \mathbb{N}$.

*Remark 1.* If Player O wins $\Gamma_{f_0}$ then she also wins $\Gamma_f$ for each $f \sqsupseteq f_0$. Analogously, if Player I wins $\Gamma_{g_0}$ then he also wins $\Gamma_g$, for each $g \sqsubseteq g_0$.

*Example 1.* Let $L \subseteq (\mathbb{B}^2)^\omega$ be given by the $\omega$-regular expression

$$\binom{0\,a}{a\,*}\Sigma^\omega + \binom{1\,*\,*\,b}{b\,*\,*\,*}\Sigma^\omega$$

where $a, b \in \mathbb{B}$ and $*$ denotes any bit. If Player I chooses 0 as his first bit then Player O needs to know $a$, so she needs delay one in this situation. Contrary, if Player I chooses 1 as his first bit then Player O needs delay three to obtain $b$. Thus, she wins the game with delay three, but neither with delay two nor one.

In the next sections we prove the following main result: *Let $\mathcal{A}$ be a DPA with $n$ states and $m$ colors, and let $n' := 2^{(mn)^{2n}}$. Then, there is a continuous operator $\lambda$ with $\binom{\alpha}{\lambda(\alpha)} \in L(\mathcal{A})$ (for all $\alpha \in \mathbb{B}^\omega$) if and only if there is a $(2n'-1)$-delay operator with the same property. To obtain this result we show that $L(\mathcal{A})$ is solvable with finite delay if and only if $L(\mathcal{A})$ is solvable with delay $2n'-1$.*

## 4 The Block Game

In this section we make the first step in the proof of our main result, which is to relax the number of bits Player I can choose in each move. For this reason we introduce a new game $\Gamma'_f$, called the *block game*.

The game $\Gamma'_f$ differs from $\Gamma_f$ in two ways. Firstly, the lengths of the words to be chosen by the players are decided by Player I, within certain intervals determined by $f$. Secondly, Player I is one move ahead compared to $\Gamma_f$.

A play in $\Gamma'_f$ is built up as follows: Player I chooses $u_0 \in \mathbb{B}^{[f(0),2f(0)]}$ and $u_1 \in \mathbb{B}^{[f(1),2f(1)]}$, then Player O chooses $v_0 \in \mathbb{B}^{|u_0|}$. In each round thereafter, i.e. for $i \geq 2$, Player I chooses $u_i \in \mathbb{B}^{[f(i),2f(i)]}$ and Player O responds by a word $v_{i-1} \in \mathbb{B}^{|u_{i-1}|}$. The winning condition is defined as before.

We show that Player I wins the game $\Gamma_f$ for all functions $f$ if and only if he wins the block game $\Gamma'_f$ for all functions $f$. To this end, for $f : \mathbb{N} \to \mathbb{N}_+$, let $f'$ be defined by $f'(0) := f(0) + f(1)$, and $f'(i) := f(i+1)$ for $i > 0$.

**Proposition 1.** *Let $f : \mathbb{N} \to \mathbb{N}_+$. If Player I wins $\Gamma_{f'}$ then he also wins $\Gamma'_f$.*

*Proof.* Assume Player I has a winning strategy in $\Gamma_{f'}$. For $i \in \mathbb{N}$, let $u_i$ be the words chosen by Player I in $\Gamma_{f'}$ and $u'_i$ the words chosen by Player I in $\Gamma'_f$, and analogously $v_i, v'_i$ for Player O. The winning strategy yields $u_0 \in \mathbb{B}^{f'(0)}$ as Player I's first move. Since $f(0) + f(1) = f'(0)$ we can choose $u'_0 u'_1 = u_0$ as

Player I's first move in $\Gamma'_f$. Player O answers by $v'_0 \in \mathbb{B}^{|u'_0|}$. We can use $v'_0$ to simulate the moves $v_0, \ldots, v_{|v'_0|-1}$ of Player O in $\Gamma_{f'}$, each of which consists of one bit. Player I answers by $u_1, \ldots, u_{|v'_0|}$ of lengths $f'(1), \ldots, f'(|v'_0|)$. Since $|v'_0| \geq 1$, the sum $f'(1)+\cdots+f'(|v'_0|)$ is non-empty and at least $f'(1) = f(2)$. Accordingly, the word $u_1 \cdots u_{|v'_0|}$ is long enough to give $u'_2$ with $f(2) \leq |u'_2| \leq 2f(2)$. We choose $u'_2$ as the prefix of $u_1 \cdots u_{|v'_0|}$ of length $f(2)$. Player O answers in $\Gamma'_f$ by $v'_1$ of length $|u'_1|$, and we can use it to simulate another $|v'_1|$ rounds in $\Gamma_{f'}$. Thereby, we obtain enough bits to give $u'_3$, and so on. This way, we build up the same plays in $\Gamma_{f'}$ and $\Gamma'_f$. Since the first one is winning for Player I, the latter one is as well. □

For $f : \mathbb{N} \to \mathbb{N}_+$, let $f''$ be inductively defined by $f''(0) := f(0)$ and

$$f''(i+1) := \sum_{j=0}^{2(f''(0)+\ldots+f''(i))} f(j).$$

**Proposition 2.** *Let $f : \mathbb{N} \to \mathbb{N}_+$. If Player I wins $\Gamma'_{f''}$ then he also wins $\Gamma_f$.*

*Proof.* Assume Player I has a winning strategy in $\Gamma'_{f''}$. For $i \in \mathbb{N}$, let $u'_i$ be the words chosen by Player I in $\Gamma'_{f''}$ and $u_i$ the words chosen by Player I in $\Gamma_f$, and analogously $v'_i, v_i$ for Player O. Player I's winning strategy yields $u'_0 \in \mathbb{B}^{[f''(0),2f''(0)]}$ and $u'_1 \in \mathbb{B}^{[f''(1),2f''(1)]}$ as his first move in $\Gamma'_{f''}$. For $i \in \mathbb{N}$, let $d'_i$ be the length of $u'_i$. Since

$$d'_0 + d'_1 \geq f''(0) + f''(1) = f(0) + \sum_{j=0}^{2f''(0)} f(j),$$

we can give the moves $u_0, \ldots, u_{d'_0}$ of Player I in $\Gamma_f$. This yields Player O's answers $v_0, \ldots, v_{d'_0-1}$, i.e. $d'_0$ bits. We can use them to simulate $v'_0$, i.e. Player O's first move in $\Gamma'_{f''}$. Player I's winning strategy yields $u'_2$ of length $f''(2) \leq d'_2 \leq 2f''(2)$. We need to give another $d'_1$ moves of Player I in $\Gamma_f$ to obtain Player O's answers $v_{d'_0}, \ldots, v_{d'_0+d'_1-1}$. For that we need $f(d'_0+1)+\ldots+f(d'_0+d'_1)$ bits. With $u'_2$ in our hands we can give these moves, because

$$\begin{aligned} d'_2 \geq f''(2) &= f(0) + \ldots + f(2f''(0) + 2f''(1)) \\ &\geq f(0) + \ldots + f(d'_0 + d'_1) \\ &\geq f(d'_0+1) + \ldots + f(d'_0 + d'_1). \end{aligned}$$

Iterating this we obtain the same plays built up in $\Gamma'_{f''}$ and $\Gamma_f$. Since Player I wins $\Gamma'_{f''}$, he also wins $\Gamma_f$. □

The following corollary of Propositions 1 and 2 is the first step in our proof.

**Corollary 1.** *Let $\mathcal{A}$ be a DPA. Then the following are equivalent:*

*(1) For all $f : \mathbb{N} \to \mathbb{N}_+$ Player I wins $\Gamma_f(\mathcal{A})$.*
*(2) For all $f : \mathbb{N} \to \mathbb{N}_+$ Player I wins $\Gamma'_f(\mathcal{A})$.*

## 5 The Semigroup Game

In this section we introduce a game which is independent of any particular delay. To define it, we extract from a DPA $\mathcal{A}$ two equivalence relations, one for each player, such that the moves of the players are classes of these relations. The first one (for Player O) is denoted $\sim$ and induces a finite semigroup on the set $(\mathbb{B}^2)^*$. The second one (for Player I) is denoted $\approx$ and ranges over $\mathbb{B}^*$. Roughly speaking, two (pairs of) words are equivalent if they effect the same behavior on $\mathcal{A}$.

Our approach to transform parity automata into finite semigroups is similar to the constructions presented in [11, 12]. Let $\mathcal{A} = (Q, q_0, \delta, c)$ be a parity automaton over $\mathbb{B}^2$. We use the semiring $\mathcal{S} := (\{-\infty\} \cup c(Q), +, \cdot)$ in which addition is defined as maximum, i.e. $x + y := \max(x, y)$ with $-\infty$ being the least element, and multiplication is defined as follows:

$$x \cdot y := \begin{cases} \max(x, y) & \text{if } x \neq -\infty \text{ and } y \neq -\infty \\ -\infty & \text{otherwise} \end{cases}$$

Note that the set $L_{\text{eq}} := (\mathbb{B}^2)^*$, i.e. the set of pairs of words of equal length, is a regular language. With each pair $\binom{u}{v} \in L_{\text{eq}}$ we associate a matrix $\mu\binom{u}{v}$ of size $|Q|^2$ with entries in $\mathcal{S}$, i.e. $\mu\binom{u}{v} \in \mathcal{S}^{Q \times Q}$, defined as follows:

$$\mu\binom{u}{v}_{p,q} := \begin{cases} -\infty & \text{if } \delta^*\left(p, \binom{u}{v}\right) \neq q \\ \max\{c(\pi)\} & \text{if } \delta^*\left(p, \binom{u}{v}\right) = q \text{ and } \pi \text{ is the associated path} \end{cases}$$

Observe that $\mathcal{S}^{Q \times Q}$ induces a finite semigroup and $\mu\binom{u}{v} \cdot \mu\binom{u'}{v'} = \mu\binom{uu'}{vv'}$. Let $\sim$ be the equivalence relation on $L_{\text{eq}}$ defined by: $\binom{u}{v} \sim \binom{u'}{v'}$ if and only if $\mu\binom{u}{v} = \mu\binom{u'}{v'}$. For each $\binom{u}{v}$, the equivalence class $\left[\binom{u}{v}\right]$ is identified by a matrix $\mu \in \mathcal{S}^{Q \times Q}$. Since $\mathcal{S}$ and $Q$ are finite, $\mathcal{S}^{Q \times Q}$ is finite as well, and so the relation $\sim$ has finite index, i.e. it has finitely many equivalence classes. We denote the index of $\sim$ by index($\sim$). Note that $L_{\text{eq}}/_\sim$ induces a finite semigroup, and $\mu$ is a semigroup morphism from $(L_{\text{eq}}/_\sim, \cdot)$ to $(\mathcal{S}^{Q \times Q}, \cdot)$.

**Lemma 3.** *Let $\binom{u}{v} \in L_{\text{eq}}$. Then, the set $\left[\binom{u}{v}\right]$ is a regular $*$-language over $\mathbb{B}^2$.*

*Proof.* We construct an automaton recognizing $\left[\binom{u}{v}\right]$ as follows: First, we construct for all $p, q \in Q, k \in c(Q)$ the automaton $\mathcal{A}_{p,q,k}$ recognizing the set of all words that induce a path from $p$ to $q$ in $\mathcal{A}$ where $k$ is the highest color seen on that path. The idea for this construction is to simulate the behavior of $\mathcal{A}$ while memorizing the highest color seen. To this end, define $\mathcal{A}_{p,q,k} := (c(Q) \times Q, \mathbb{B}^2, (c(p), p), \delta', \{(k, q)\})$ where

$$\delta'\left((k', p'), \binom{x}{y}\right) := \left(\max\left\{k', c\left(\delta\left(p', \binom{x}{y}\right)\right)\right\}, \delta\left(p', \binom{x}{y}\right)\right)$$

for all $k' \in c(Q), p' \in Q, x, y \in \mathbb{B}$. The automaton starts in the state $(c(p), p)$ and simulates the behavior of $\mathcal{A}$ on its input. If it stops in state $(k, q)$ then it accepts. The automaton $\mathcal{A}_{[\binom{u}{v}]}$ is then obtained as the intersection of all $\mathcal{A}_{p,q,k}$ for $p, q, k$ such that $\mu\binom{u}{v}_{p,q} = k$. □

Since $\sim$ has finite index, we can find automata for all equivalence classes of $\sim$ in the following way. For $r \in \mathbb{N}$, let $\mathcal{A}_1, \ldots, \mathcal{A}_r$ be the automata already constructed. Then $\sim$ has index $r$ if and only if $\bigcup_{i=1,\ldots,r} L(\mathcal{A}_i) = L_{\mathrm{eq}}$. This equality can be effectively checked, and if this test fails, we repeat the construction with a word contained in $L_{\mathrm{eq}} \setminus \bigcup_{i=1,\ldots,r} L(\mathcal{A}_i)$.

Let $\approx$ be the equivalence relation on $\mathbb{B}^*$ defined by

$$u \approx u' :\iff \forall\left[\binom{u_0}{v_0}\right] : \left(\exists v : \binom{u}{v} \in \left[\binom{u_0}{v_0}\right] \iff \exists v' : \binom{u'}{v'} \in \left[\binom{u_0}{v_0}\right]\right).$$

For $u \in \mathbb{B}^*$, the $\approx$-equivalence class of $u$, denoted $[u]$, can be identified with a subset of all $\sim$-classes. Since $\sim$ has finite index, we get that $\approx$ has finite index as well; more precisely it holds that $\mathrm{index}(\approx) \leq 2^{\mathrm{index}(\sim)}$.

**Lemma 4.** *Let $u \in \mathbb{B}^*$. Then, the set $[u]$ is a regular $*$-language over $\mathbb{B}$.*

*Proof.* We construct an automaton recognizing the language $[u]$ as follows: First, we have to check for which $\sim$-classes $\left[\binom{u_0}{v_0}\right]$ there exists $v \in \mathbb{B}^{|u|}$ such that $\binom{u}{v} \in \left[\binom{u_0}{v_0}\right]$. Let $\mathcal{B}$ be a DFA recognizing $\left[\binom{u_0}{v_0}\right]$. We take the projection on the first component (deleting the second component from the transitions of $\mathcal{B}$) and test whether the resulting automaton, say $\mathcal{B}'$, accepts $u$. If we do the same for all $\sim$-classes, then we obtain $r$ automata $\mathcal{B}'_1, \ldots, \mathcal{B}'_r$ accepting $u$, and $s$ automata $\mathcal{B}'_{r+1}, \ldots, \mathcal{B}'_{r+s}$ not accepting $u$, where $r + s = \mathrm{index}(\sim)$. From these automata we can effectively construct an automaton for $[u]$, because

$$[u] = \bigcap_{i=1,\ldots,r} L(\mathcal{B}'_i) \cap \bigcap_{j=r+1,\ldots,r+s} \overline{L(\mathcal{B}'_j)}.$$

□

We now define the game $\Gamma^{SG}$ (induced by a DPA $\mathcal{A}$ over $\mathbb{B}^2$) where the moves of the players are classes from $\mathbb{B}^*/_{\approx}$ and $L_{\mathrm{eq}}/_{\sim}$, respectively. Accordingly, we call $\Gamma^{SG}$ the *semigroup game* of $\mathcal{A}$.

The game $\Gamma^{SG}$ is defined similar to the block game $\Gamma'$. The difference is that the players do not choose concrete words but the respective classes from the relations $\sim$ and $\approx$. A play is built up as follows. Player I chooses infinite classes $[u_0], [u_1] \in \mathbb{B}^*/_{\approx}$, then Player O chooses a class $\left[\binom{u_0}{v_0}\right] \in L_{\mathrm{eq}}/_{\sim}$. In each round thereafter, i.e. for $i \geq 2$, Player I chooses an infinite class $[u_i] \in \mathbb{B}^*/_{\approx}$ and Player O chooses a class $\left[\binom{u_{i-1}}{v_{i-1}}\right] \in L_{\mathrm{eq}}/_{\sim}$. A play is winning for Player O if and only if $\binom{u_0}{v_0}\binom{u_1}{v_1}\binom{u_2}{v_2}\cdots$ is accepted by $\mathcal{A}$.

Note that $\mathbb{B}^*/_{\approx}$ contains at least one infinite class and that for each class $[u]$ there exists at least one class in $L_{\mathrm{eq}}/_{\sim}$ associated with $[u]$ (by the definition of $\approx$). Hence, both players can always move. Furthermore, the winning condition of $\Gamma^{SG}$ is well-defined because acceptance of $\mathcal{A}$ is independent of representatives: If $\left[\binom{u_i}{v_i}\right] = \left[\binom{u'_i}{v'_i}\right]$ for all $i \in \mathbb{N}$, then $\binom{u_0}{v_0}\binom{u_1}{v_1}\cdots \in L(\mathcal{A}) \iff \binom{u'_0}{v'_0}\binom{u'_1}{v'_1}\cdots \in L(\mathcal{A})$.

$\Gamma^{SG}$ can be modeled by a parity game on a finite graph. (Thus, the winner of $\Gamma^{SG}$ is computable, using standard results on parity games [1].) In the vertices we keep track of the current state of $\mathcal{A}$ and the $\approx$-classes chosen by Player I. The vertex reached by a move $\left[\binom{u}{v}\right]$ of Player O is colored by $\mu\binom{u}{v}_{q,q'}$, where $q, q'$ are the states reached in $\mathcal{A}$ before and after reading $\binom{u}{v}$.

## 6 Connecting the Block and the Semigroup Game

In this section we show that Player I wins the block game $\Gamma'_f$ for all functions $f : \mathbb{N} \to \mathbb{N}_+$ if and only if he wins the semigroup game $\Gamma^{SG}$. This completes the reduction and also yields the proof of our main result.

The basic idea of the proof of Theorem 1 (see below) is, for arbitrary $f$, to simulate the players' moves in $\Gamma'_f$ by the corresponding classes of the relations $\sim$ and $\approx$, respectively, and vice versa. For the direction from left to right, one has the problem whether a class $[u_i]$ contains an appropriate representative, i.e. one of length between $f(i)$ and $2f(i)$. We use Lemma 1 to show that there exists a particular $f$ such that each function $g$ with $g \sqsupseteq f$ indeed has this property. Then, the following lemma completes the proof.

**Lemma 5.** *Player I wins $\Gamma'_f$ for all functions $f : \mathbb{N} \to \mathbb{N}_+$ if and only if there exists a function $f_0 : \mathbb{N} \to \mathbb{N}_+$ such that Player I wins $\Gamma'_g$ for all $g \sqsupseteq f_0$.*

*Proof.* The direction from left to right is immediate. Conversely, assume there exists $f_0$ such that Player I does not win $\Gamma'_{f_0}$. Determinacy yields that Player O wins $\Gamma'_{f_0}$. By Proposition 1 Player O wins $\Gamma_{f'_0}$, and from Remark 1 it follows that she also wins $\Gamma_f$ for all $f \sqsupseteq f'_0$. Proposition 2 yields that Player O wins $\Gamma'_{f''}$, for all $f \sqsupseteq f'_0$. Towards a contradiction, let $f_+$ be a function such that Player I wins $\Gamma'_g$ for all $g \sqsupseteq f_+$, and let $f_*$ be the maximum of $f_+$ and $f'_0$, i.e. for all $i \in \mathbb{N}$

$$f_*(i) := \max\{f_+(i), f'_0(i)\}.$$

Since $f_* \sqsupseteq f'_0$ it holds that Player O wins $\Gamma'_{f''_*}$. However, since $f''_* \sqsupseteq f_* \sqsupseteq f_+$ Player I must win $\Gamma'_{f''_*}$, by assumption. This yields a contradiction which means that $f_+$ cannot exist. □

Lemma 5 and the next theorem establish the second step of our reduction.

**Theorem 1.** *Player I wins $\Gamma^{SG}$ if and only if there is a function $f : \mathbb{N} \to \mathbb{N}_+$ such that Player I wins $\Gamma'_g$ for all $g \sqsupseteq f$.*

*Proof.* We start with the direction from right to left. Let $f : \mathbb{N} \to \mathbb{N}_+$ be a function such that Player I wins $\Gamma'_g$ for all $g \sqsupseteq f$. We define a function $g$ such that $g \sqsupseteq f$ and each word of length $g(i)$ is contained in an infinite $\approx$-class, for all $i \in \mathbb{N}$. To this end, let $d'$ be the length of a longest word in all finite $\approx$-classes[3] and define, for all $i \in \mathbb{N}$, $g(i) := \max\{f(i), d' + 1\}$.

Player I wins $\Gamma'_g$ by assumption, and a winning strategy yields his first two moves $u_0, u_1$. Both $[u_0]$ and $[u_1]$ are infinite, and so he can choose them in $\Gamma^{SG}$. We simulate Player O's answer $\left[\binom{u_0}{v_0}\right]$ by choosing $v_0$ in $\Gamma'_g$, and obtain $u_2$ with $[u_2]$ being infinite. Choosing $[u_2]$ in $\Gamma^{SG}$ we obtain Player O's next move $\left[\binom{u_1}{v_1}\right]$, and so on.

We argue that the plays built up have the same maximal color occurring infinitely often. It suffices to show that in both plays a move of Player O leads

[3] If $\approx$ has no finite equivalence class, then we define $d' := 0$.

$\mathcal{A}$ to the same state, via paths with equal maximal color. Then, the rest follows by induction. Let $q_i$ be the current state of $\mathcal{A}$ and $u_i, u_{i+1}$ be the words chosen by Player I. If Player O chooses $\left[\binom{u_i}{v_i}\right]$ in $\Gamma^{SG}$, then we reach the state $q_{i+1} := \delta^*\big(q_i, \binom{u_i}{v_i}\big)$ via the maximal color $\mu\binom{u_i}{v_i}_{q_i,q_{i+1}}$. The state $q_{i+1}$ is well-defined because from $q_i$ every $\binom{u_i'}{v_i'} \in \left[\binom{u_i}{v_i}\right]$ leads $\mathcal{A}$ to the same state, though via different paths, but with the same maximal color. In $\Gamma_g'$ Player O chooses $v_i$. As in $\Gamma^{SG}$, we reach the state $q_{i+1}$ via the maximal color $\mu\binom{u_i}{v_i}_{q_i,q_{i+1}}$.

Conversely, assume that Player I wins $\Gamma^{SG}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_r$ be automata recognizing all the $\approx$-classes, and $n'$ the maximal number of states among these automata, i.e. $n' := \max\{n_1, \ldots, n_r\}$, where $n_j$ is the number of states of $\mathcal{A}_j$ $(j = 1, \ldots, r)$. We define the constant function $f$ by $f(i) := n'$ for all $i \in \mathbb{N}$, and show that Player I wins $\Gamma_g'$ for all $g$ with $g \sqsupseteq f$. Player I's winning strategy in $\Gamma^{SG}$ yields $[u_0], [u_1]$. Since $[u_0], [u_1]$ are infinite, we can apply Lemma 1. Accordingly, each $\mathcal{A}_j$ accepts a word of length between $f$ and $f + n_j$ and hence between $f$ and $2f$ because $n_j \leq n' \leq f$.[4] Hence, we can assume w.l.o.g. that $f \leq |u_0|, |u_1| \leq 2f$. Player I chooses $u_0, u_1$ in $\Gamma_f'$ and Player O answers by a word $v_0$ with $|v_0| = |u_0|$. We simulate this move by $\left[\binom{u_0}{v_0}\right]$ in $\Gamma^{SG}$ and obtain Player I's answer $[u_2]$, so the next move of Player I in $\Gamma_f'$ is $u_2$ (for appropriate $u_2$). Player O chooses $v_1$ with $|v_1| = |u_1|$, and so on.

The plays built up this way have the same maximal color occurring infinitely often, using the same inductive argument as above. Starting at $q_i$, Player O's move $v_i$ in $\Gamma_f'$ has the same effect as the corresponding move $\left[\binom{u_i}{v_i}\right]$ in $\Gamma^{SG}$, i.e. we reach the state $q_{i+1} := \delta^*\big(q_i, \binom{u_i}{v_i}\big)$ via the maximal color $\mu\binom{u_i}{v_i}_{q_i,q_{i+1}}$.

What remains to be shown is that Player I wins $\Gamma_g'$ for all $g \sqsupseteq f$. Let $|[a, b]| := b - a$ be the size of the interval $[a, b]$. If $g \sqsupseteq f$ then, since $|[f, 2f]| = n'$, it holds that $|[g(i), 2g(i)]| \geq n'$ for all $i \in \mathbb{N}$. Hence, to win $\Gamma_g'$ Player I simply chooses longer representatives of the $\approx$-classes than in $\Gamma_f'$. □

A thorough analysis of the constructions of the $\sim$-classes and $\approx$-classes, respectively, yields an upper bound for $n'$. Let $n$ be the number of states of $\mathcal{A}$ and $m$ the number of colors. Let $u, v \in \mathbb{B}^*$ with $|u| = |v|$. Since $\mathcal{A}$ is deterministic, there is exactly one entry distinct from $-\infty$ in each of the $n$ rows of $\mu\binom{u}{v}$, and $\mathcal{A}_{p,q,k}$ has at most $mn$ states. Hence, each $\mathcal{A}_{[\binom{u}{v}]}$ has at most $(mn)^n$ states, i.e. as many as the product of $n$ (deterministic) automata of size $mn$. To obtain an automaton for a class $[u]$ we have to intersect index($\sim$) languages (cf. page 10). By the same argument as above, there are at most $(mn)^n$ possible matrices identifying all the $\sim$-classes. Since our construction includes determinization, we obtain each $\mathcal{A}_{[u]}$ having at most $k$ states, where

$$k \leq (2^{(mn)^n})^{(mn)^n} = 2^{(mn)^{2n}}.$$

Next, we obtain our main result showing that in regular games constant delay is sufficient for Player O to win, if she can win with delay at all.

[4] To simplify matters we write $f$ instead of $f(i)$.

**Lemma 6.** *Let $n'$ be as in the proof of Theorem 1. Then, Player O wins $\Gamma^{SG}$ if and only if Player O wins $\Gamma_{\langle 2n'-1\rangle}$.*

*Proof.* Define $f(i) := n'$ for all $i \in \mathbb{N}$ and let $w$ of length $d'$ be a longest word in all finite $\approx$-classes. Moreover, let $L(\mathcal{A}') = [w]$, where $\mathcal{A}'$ has $n$ states. Then, we have $d' < n$. Otherwise, the run of $\mathcal{A}'$ on $w$ would have a loop, which is a contradiction to the finiteness of $L(\mathcal{A}')$. Since $n \leq n'$ we get $d' < n'$ and so $d' + 1 \leq n'$. Thus, each $\approx$-class containing a word of length at least $f$ is infinite.

Assume that Player O wins $\Gamma^{SG}$. We first show that Player O wins $\Gamma'_f$. Let $u_0, u_1$ with $n' \leq |u_0|, |u_1| \leq 2n'$ be the first move of Player I in $\Gamma'_f$. By the above remarks $[u_0], [u_1]$ are infinite, and we can simulate $[u_0], [u_1]$ in $\Gamma^{SG}$. Player O's winning strategy in $\Gamma^{SG}$ yields $\left[\binom{u_0}{v_0}\right]$ for some suitable $v_0$. Let him choose $v_0$ in $\Gamma'_f$. Then Player I chooses $u_2$ and we simulate $[u_2]$ in $\Gamma^{SG}$, and so on.

As in the proof of Theorem 1, we obtain plays with the same maximal color occurring infinitely often, and so Player O wins $\Gamma'_f$. Simulating a winning strategy for $\Gamma'_f$ she also wins $\Gamma_{\langle 2n'-1\rangle}$. The factor 2 comes from the fact that we need at least $2n'$ bits when simulating Player I's first move in $\Gamma'_f$.

Conversely, let Player O win $\Gamma_{\langle 2n'-1\rangle}$ and $g(i) := 2n'$, for all $i \in \mathbb{N}$. Since $g \sqsupseteq \langle 2n'-1\rangle$, Player O wins $\Gamma_g$. Then, by Proposition 2, Player O also wins $\Gamma'_{g''}$. Given a winning strategy for Player O in $\Gamma'_{g''}$ we can specify one for her in $\Gamma^{SG}$ as follows: A move $[u_i]$ of Player I is simulated by $u_i$ in $\Gamma'_{g''}$, for $g''(i) \leq |u_i| \leq 2g''(i)$. (By Lemma 1, an appropriate representative $u_i$ must exist because $g'' \sqsupseteq g$, and so $|[g''(i), 2g''(i)]| \geq n'$ for all $i \in \mathbb{N}$.) We use Player O's answer $v_{i-1}$ to choose $\left[\binom{u_{i-1}}{v_{i-1}}\right]$ in $\Gamma^{SG}$. This yields a play winning for Player O in $\Gamma^{SG}$. □

**Theorem 2.** *Let $\mathcal{A}$ be a DPA over $\mathbb{B}^2$. Then, $L(\mathcal{A})$ is solvable with finite delay if and only if $L(\mathcal{A})$ is solvable with delay $2n'-1$. There is a continuous operator $\lambda$ such that $\{\binom{\alpha}{\lambda(\alpha)} \mid \alpha \in \mathbb{B}^\omega\} \subseteq L(\mathcal{A})$ if and only if there is a $(2n'-1)$-delay operator with the same property.*

In a game of constant delay $d$ the number of different bit sequences Player I can move ahead is globally bounded. The game $\Gamma_{\langle d\rangle}(\mathcal{A})$ can be modeled by a parity game on a finite graph of size at most $2^{d+1} \cdot |\mathcal{A}|$. By standard techniques [1], such games can be solved in time $O((2^{d+1} \cdot n)^m)$.

**Corollary 2.** *Let $\mathcal{A}$ be a DPA over $\mathbb{B}^2$. The problem whether $L(\mathcal{A})$ is solvable with finite delay and the problem whether there is a continuous operator $\lambda$ with $\{\binom{\alpha}{\lambda(\alpha)} \mid \alpha \in \mathbb{B}^\omega\} \subseteq L(\mathcal{A})$ are in* 3ExpTime.

## 7 Lookahead in Non-Regular Games

In this section we show that the above results do not hold for context-free $\omega$-languages ($\mathrm{CFL}_\omega$, for an introduction see e.g. [13]). Let us first recall that it is undecidable whether a context-free $\omega$-language $L \subseteq \mathbb{B}^\omega$ is universal, i.e. whether $L = \mathbb{B}^\omega$ holds.

**Theorem 3 (see also [14]).** *Let $L \subseteq (\mathbb{B}^2)^\omega$ be a context-free $\omega$-language. Then, it is undecidable whether there exists $f$ such that Player O wins $\Gamma_f(L)$.*

*Proof.* We make a reduction from the universality problem for context-free $\omega$-languages. Let $L_I \in \mathrm{CFL}_\omega$ and $L := \left\{ \binom{\alpha}{\beta} \mid \alpha \in L_I, \beta \in \mathbb{B}^\omega \right\}$. If $L_I$ is universal then $L$ is universal as well, and Player O wins with any $f$. Conversely, if $L_I$ is not universal, then Player I wins by playing a word $\alpha \notin L_I$. There is no response $\beta$ such that $\binom{\alpha}{\beta} \in L$, therefore Player O looses with each $f$. Altogether, $L_I$ is universal if and only if there exists $f$ such that Player O wins $\Gamma_f(L)$. □

The situation is different for *deterministic* $\omega$-context-free specifications: in this case at least the winner of the standard game $\Gamma_{\langle 0 \rangle}$ is decidable [3].

In addition to undecidability for the general case, we show that there exist context-free specifications which are solvable with finite delay, but not with constant delay.

*Example 2.* Let $L \subseteq (\mathbb{B}^2)^\omega$ be defined such that if Player I chooses an $\omega$-word of the form $\alpha = 1^{2m_0}0^{n_0}1^{2m_1}0^{n_1}\cdots$, for $m_i, n_i \in \mathbb{N}_+$, then Player O wins if and only if he answers by $\beta = 1^{m_0}0^{m_0+n_0}1^{m_1}0^{m_1+n_1}\cdots$. This means Player O's $i$th block of 1s must have exactly half the length of Player I's $i$th block of 1s, and both blocks must start at the same position. If $\alpha$ is not of the above form, then Player O wins as well.

The language $L$ is recognized by a deterministic $\omega$-pushdown automaton. As long as the input is $\binom{1}{1}$, we push a symbol on the stack. If we read the first $\binom{1}{0}$ after $\binom{1}{1}$, we start to pop symbols from the stack. If we reach the initial stack symbol at the same time as we read the first $\binom{0}{0}$ after $\binom{1}{0}$ then we are satisfied and visit a final state.

Observe that Player O wins $\Gamma_f(L)$, if $f(i) := 2$ for all $i \in \mathbb{N}$. When she has to give her $i$th bit $\beta_i$ she already knows Player I's $(2i)$th bit $\alpha_{2i}$, and that is enough to decide whether to play 0 or 1.

Let us show that $L$ is not solvable with constant delay. Towards a contradiction, assume Player O wins $\Gamma_{\langle d \rangle}$ for some $d \in \mathbb{N}$. We construct a winning strategy for Player I in $\Gamma_{\langle d \rangle}$ as follows: Player I chooses $1^{d+1}$ as initial move and afterwards $d$ further 1s. Player O must answer by choosing $(d+1)$ 1s. Afterwards, Player I has to choose another 1 to complete his block of 1s to even length. (After this move, Player I has chosen exactly twice as many 1s as Player O.) Whatever Player O answers, say $b$, Player I wins by choosing $1-b$ next. This is due to the fact that the block of 1s chosen by Player O gets either too short or too long.

## 8 Conclusion

In this paper we introduced and compared strategies with different kinds of lookahead in regular infinite games. We showed that continuous strategies can be reduced to uniformly continuous strategies of a special form, namely strategies with bounded lookahead. This result is a first step into a wider – and it seems

rather unexplored – topic. Let us mention some aspects. First, it is straightforward to present the results in a set-up that is symmetric in the two players. We also skipped here a lower bound proof for the double exponential size in Theorem 2. It is also possible to think of "infinite lookahead" where, for instance, the second player may use information about the first player's sequence up to a partition of the space of sequences into regular sets.

We showed that bounded lookahead is not enough for continuous strategies in non-regular games. It is open which functions may be appropriate for uniformly continuous operators in such games. An appropriate framework is given by the deterministic context-free $\omega$-languages, as the class of all context-free $\omega$-languages is too wide for effective results. One could conjecture that in deterministic context-free games, polynomial functions are sufficient as bounds for uniformly continuous strategies. Also, it is open whether solvability with continuous or uniformly continuous strategies is decidable for such games.

## References


[1] Grädel, E., Thomas, W., Wilke, T., eds.: Automata, Logics and Infinite Games. Volume 2500 of LNCS. Springer (2002)

[2] Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the AMS **138** (1969) 295–311

[3] Walukiewicz, I.: Pushdown processes: Games and model checking. In Alur, R., Henzinger, T.A., eds.: CAV. Volume 1102 of LNCS., Springer (1996) 62–74

[4] Cachat, T.: Higher order pushdown automata, the caucal hierarchy of graphs and parity games. In Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J., eds.: ICALP. Volume 2719 of LNCS., Springer (2003) 556–569

[5] Bouquet, A.J., Serre, O., Walukiewicz, I.: Pushdown games with unboundedness and regular conditions. In Pandya, P.K., Radhakrishnan, J., eds.: FSTTCS. Volume 2914 of LNCS., Springer (2003) 88–99

[6] Moschovakis, Y.N.: Descriptive Set Theory. Volume 100 of Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Company (1980)

[7] Trakhtenbrot, B.A., Barzdin, Y.M.: Finite Automata, Behavior and Synthesis. North Holland, Amsterdam (1973)

[8] Thomas, W., Lescow, H.: Logical specifications of infinite computations. In de Bakker, J.W., de Roever, W.P., Rozenberg, G., eds.: REX School/Symposium. Volume 803 of LNCS., Springer (1993) 583–621

[9] Hosch, F.A., Landweber, L.H.: Finite delay solutions for sequential conditions. In Nivat, M., ed.: Automata, Languages and Programming, Paris, France, North-Holland, Amsterdam (1972) 45–60

[10] Even, S., Meyer, A.: Sequential boolean equations. IEEE Transactions on Computers **C-18** (1969) 230–240

[11] Perrin, D., Pin, J.: Semigroups and automata on infinite words. In Fountain, J., ed.: NATO Advanced Study Institute *Semigroups, Formal Language and Groups.* Kluwer academic publishers (1995) 49–72

[12] Pin, J.: Finite semigroups and recognizable languages: An introduction (1995)

[13] Cohen, R.S., Gold, A.Y.: Omega-computations on deterministic pushdown machines. Journal of Computer and System Sciences **16** (1978) 275–300

[14] Finkel, O.: Topological properties of omega context-free languages. Theoretical Computer Science **262** (2001) 669–697