Degrees of Lookahead in Context-free Infinite Games*

Wladimir Fridman, Christof Löding, and Martin Zimmermann

Lehrstuhl Informatik 7, RWTH Aachen University, Germany
{fridman,loeding,zimmermann}@automata.rwth-aachen.de

Abstract

We continue the investigation of delay games, infinite games in which one player may postpone her moves for some time to obtain a lookahead on her opponent's moves. We show that the problem of determining the winner of such a game is undecidable for deterministic context-free winning conditions. Furthermore, we show that the necessary lookahead to win a deterministic context-free delay game cannot be bounded by any elementary function. Both results hold already for restricted classes of deterministic context-free winning conditions.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs; F.4.3 Formal Languages

Keywords and phrases Infinite games, delay, context-free languages

Digital Object Identifier 10.4230/LIPIcs.CSL.2011.264

1 Introduction

Many of today's problems in computer science are no longer concerned with programs that transform data and then terminate, but with non-terminating reactive systems which have to interact with an (possibly) antagonistic environment for an unbounded amount of time. The framework of infinite two-player games is a powerful and flexible tool to verify and synthesize such systems [6]. The seminal theorem of Büchi and Landweber [2] states that the winner of an infinite game on a finite arena with a regular winning condition can be determined and a corresponding finite-state winning strategy can be constructed effectively.

Ever since, this result was extended along different dimensions, e.g., the number of players, the type of arena, the type of winning condition, the type of interaction between the players (alternation or concurrency), zero-sum or nonzero-sum, and complete or incomplete information. In this work, we consider two of these dimensions, namely context-free winning conditions and the possibility for one player to delay her moves.

Walukiewicz showed that games with deterministic context-free winning conditions can be solved in exponential time [12]. On the other hand, the problem of determining the winner of a game with (non-deterministic) context-free winning condition is undecidable, which can be shown by a reduction from the universality problem for non-deterministic ω-pushdown automata (see, e.g. [5]).

In a delay game, one of the players can postpone her moves for some time, thereby obtaining a lookahead on the moves of her opponent. This allows her to win some games which she loses without lookahead, e.g., if her first move depends on the third move of her opponent. On the other hand, there are simple winning conditions that cannot be won with

* The third author was supported by the project *Games for Analysis and Synthesis of Interactive Computational Systems (GASICS)* of the *European Science Foundation*.

Computer Science Logic 2011 (CSL'11).
Editor: Marc Bezem; pp. 264–276
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

any finite lookahead, e.g., if her first move depends on all of the infinitely many moves of her opponent. Delay arises naturally when transmission of data in networks or components equipped with buffers are modeled. Also, from a theoretical point of view, uniformization of relations by continuous functions can be expressed and analyzed in this setting.

Hosch and Landweber proved that it is decidable, whether a game with regular winning condition can be won with bounded lookahead (i.e., only finitely many moves are postponed) [8]. This result was improved by Holtmann, Kaiser, and Thomas [7] who showed that if a player wins a game with arbitrary lookahead, then already with (doubly-exponential) bounded lookahead, and gave a streamlined decidability proof.

We consider games in which two players pick letters from alphabets Σ_I and Σ_O, respectively, thereby producing two infinite sequences α and β. Thus, a strategy for the second player induces a mapping $\sigma\colon \Sigma_I^\omega \to \Sigma_O^\omega$. It is winning for the second player if $(\alpha, \sigma(\alpha))$ is contained in the winning condition $L \subseteq \Sigma_I^\omega \times \Sigma_O^\omega$ for every α. In this case, we say that σ uniformizes L. In the classical setting, in which the players pick letters in alternation, the n-th letter of $\sigma(\alpha)$ depends only on the first n letters of α. A strategy with bounded lookahead induces a Lipschitz-continuous function σ (in the Cantor topology on Σ^ω) and a strategy with arbitrary lookahead induces a continuous function (or equivalently, a uniformly continuous function, as Σ^ω is compact).

Thus, stated in these terms, Hosch and Landweber proved the decidability of the uniformization problem for regular relations by Lipschitz-continuous functions. Holtmann, Kaiser, and Thomas proved the equivalence of the existence of a continuous uniformization function and the existence of a Lipschitz-continuous uniformization function for regular relations. They observe that this equivalence does not hold for deterministic context-free winning conditions by giving an example in which every other move has to be postponed, i.e., the lookahead grows linearly. They ask whether the winner of such a game can be determined effectively and what kind of lookahead is necessary to win. We answer these questions.

Firstly, by applying the result of Walukiewicz [12] it is easy to see that if we only allow a fixed bounded lookahead, then determining the winner is decidable. Then, we show that determining whether a given player wins the game with arbitrary lookahead is undecidable for deterministic context-free winning conditions. We complement this by giving a criterion to determine when this question is decidable, if the lookahead is restricted to some fixed class of functions. Intuitively, if there is no global bound on the lookahead provided by the class, then the problem of determining the winner is undecidable. If there is such a bound, then it follows from the decidability result that the winner can be determined effectively.

By closely inspecting the winning conditions constructed in the proofs, it follows that these undecidability results already hold for winning conditions given by visibly one-counter automata. These are pushdown automata that have a single stack symbol, i.e., their stack is essentially a counter which can be incremented, decremented, and tested for zero, and whose input letters control the behavior of the stack, i.e., a letter either always triggers a push transition, a pop transition, or a skip transition (the stack is not changed). Visibly pushdown automata are a popular choice to model recursive processes as their languages are closed under all boolean operations (as opposed to (deterministic) context-free languages in general), they can be determinized, and have good algorithmic properties (for a thorough discussion see [1], for the determinization procedure for visibly pushdown automata on ω-words see [9]). At the same time, we do not need the full strength of the parity or Muller acceptance condition: all winning conditions can be recognized by automata with weak acceptance conditions that refer only to the set of visited states, and not to the set of states visited infinitely often.

Finally, we consider the lookahead necessary to win delay games with deterministic context-free winning conditions. We present a deterministic context-free delay game which can be won if arbitrary lookahead is available, but not with lookahead that is bounded by an elementary function, i.e., bounded by a k-fold exponential for some fixed k. Again, the winning condition can be recognized by a visibly one-counter automaton with weak acceptance condition.

In terms of uniformization of relations by continuous functions, we show that it is undecidable to determine whether a deterministic context-free relation is uniformized by some continuous function and to determine whether it is uniformized by some Lipschitz-continuous function. Furthermore, the example by Holtmann, Kaiser, and Thomas shows that the equivalence between the existence of a continuous uniformization function and the existence of a Lipschitz-continuous uniformization function does not hold for deterministic context-free relations, as opposed to the regular case.

Our results show that, unlike in the regular case, adding lookahead to deterministic context-free games significantly changes their algorithmic properties. Bounded lookahead, which is sufficient to win regular games, can always be encoded into the winning condition, hence the classical algorithms to solve regular games without lookahead are still applicable. However, in case of deterministic context-free games, where unbounded lookahead is necessary, one cannot encode it into the winning condition while preserving its context-freeness. This is a reason why these games are hard to handle algorithmically.

This work is structured as follows: in Section 2, we introduce infinite games with delay formally and present the types of pushdown automata we consider. Then, in Section 3 we present the decidability and undecidability results. The lower bound for the lookahead is presented in Section 4. In Section 5, we conclude with some open questions.

2 Definitions

The set of non-negative integers is denoted by $\mathbb{N}$, and we define $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. For an integer $n > 0$ let $[n]$ denote the set $\{0, \dots, n-1\}$. We define the k-fold exponential function $\exp_k \colon \mathbb{N} \to \mathbb{N}$ inductively by $\exp_0(n) = n$, and $\exp_{k+1}(n) = 2^{\exp_k(n)}$. An alphabet Σ is a non-empty finite set of letters, Σ^* denotes the set of finite words over Σ, Σ^n denotes the set of words over Σ of length n, and Σ^ω denotes the set of infinite words over Σ. The empty word is denoted by ε. For $\alpha \in \Sigma^* \cup \Sigma^\omega$ and $n \in \mathbb{N}$ we write $\alpha(n)$ for the n-th letter of α. For $w \in \Sigma^*$ and $a \in \Sigma$, $|w|_a$ denotes the number of a's in w.

2.1 Games with Delay

Given a *delay function* $f \colon \mathbb{N} \to \mathbb{N}_+$ and an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$, the game $\Gamma_f(L)$ is played by two players (the input player I and the output player O) in rounds $i = 0, 1, 2, \dots$ as follows: in round i, Player I picks a word $u_i \in \Sigma_I^{f(i)}$, then Player O picks one letter $v_i \in \Sigma_O$. We refer to the sequence $(u_0, v_0), (u_1, v_1), (u_2, v_2), \dots$ as a *play* of $\Gamma_f(L)$, which yields two infinite words $\alpha = u_0 u_1 u_2 \cdots$ and $\beta = v_0 v_1 v_2 \cdots$. Player O wins the play if and only if the *induced word* $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ is in L.

Given a delay function f, a *strategy* for Player I is a mapping $\tau_I \colon \Sigma_O^* \to \Sigma_I^*$ such that $|\tau_I(w)| = f(|w|)$, and a strategy for Player O is a mapping $\tau_O \colon \Sigma_I^* \to \Sigma_O$. Consider a play $(u_0, v_0), (u_1, v_1), (u_2, v_2), \dots$ of $\Gamma_f(L)$. Such a play is *consistent* with τ_I, if $u_i = \tau_I(v_0 \cdots v_{i-1})$ for every i; it is consistent with τ_O, if $v_i = \tau_O(u_0 \cdots u_i)$ for every i. A strategy τ for Player p is *winning* for her, if every play that is consistent with τ is won by Player p. In this case, we say Player p *wins* $\Gamma_f(L)$.

For a delay function $f\colon \mathbb{N} \to \mathbb{N}_+$ define its *distance function* d_f by $d_f(i) = \left(\sum_{j=0}^{i} f(j)\right) - (i+1)$, i.e., $d_f(i)$ is the lookahead attained by Player O after i rounds. We say that f is a *constant-delay function* with delay d, if $d_f(i) = d$ for all i; f is a *linear-delay function* with delay $k > 0$, if $d_f(i) = (i+1)(k-1)$ for all i; and we say that f is an *elementary-delay function*, if $d_f \in \mathcal{O}(\exp_k)$ for some fixed k. Here, "constant", "linear", and "elementary" refer to the lookahead for Player O, i.e., to the kind of distance function d_f, and not to the kind of delay function f.

▶ **Example 1.** Consider the language L over $\{0,1,\sharp\} \times \{0,1,\sharp\}$ containing the words of the form $\binom{0}{0}^{n_0}\binom{0}{1}^{n_0}\binom{\sharp}{\sharp}\binom{0}{0}^{n_1}\binom{0}{1}^{n_1}\binom{\sharp}{\sharp}\binom{0}{0}^{n_2}\binom{0}{1}^{n_2}\binom{\sharp}{\sharp}\cdots$ with $n_i > 0$ for all i, as well as all words whose first component is not of the form $0^{2n_0}\sharp 0^{2n_1}\sharp 0^{2n_2}\sharp\cdots$ with $n_i > 0$ for all i. In order to win a game with winning condition L, Player I has to produce infinitely many blocks of 0's, all of even length. If he does this, then Player O can still guarantee to win by answering the first half of every block by 0 and the second half by 1. To do so, she needs to know in which half of a block she currently is, which is guaranteed by the linear-delay function with delay 2.

2.2 Pushdown Automata

A *deterministic pushdown machine* (DPDM) is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{in}, \bot)$ where Q is a finite set of states, Σ is an input alphabet, Γ is a pushdown alphabet, $\bot \notin \Gamma$ is the initial pushdown symbol (let $\Gamma_\bot = \Gamma \cup \{\bot\}$), $q_{in} \in Q$ is the initial state, and $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma_\bot \to Q \times \Gamma_\bot^*$ is a partial transition function satisfying for every $q \in Q$ and every $A \in \Gamma_\bot$: either $\delta(q,a,A)$ is defined for all $a \in \Sigma$ and $\delta(q,\varepsilon,A)$ is undefined, or $\delta(q,\varepsilon,A)$ is defined and $\delta(q,a,A)$ is undefined for all $a \in \Sigma$. We require that the initial pushdown symbol $\bot$ can neither be written on the stack nor be deleted from the stack.

A stack content is a word from $\Gamma^*\bot$, we assume the leftmost symbol to be the top of the stack. A *configuration* is a pair (q,γ) consisting of a state $q \in Q$ and a stack content $\gamma \in \Gamma^*\bot$. We write $(q, A\gamma) \overset{a}{\vdash} (q', \gamma'\gamma)$, if $(q',\gamma') = \delta(q,a,A)$ for $a \in \Sigma \cup \{\varepsilon\}$, $\gamma, \gamma' \in \Gamma_\bot^*$ and $A \in \Gamma_\bot$. For an ω-word $\alpha \in \Sigma^\omega$ an infinite sequence of configurations $\rho = (q_0,\gamma_0)(q_1,\gamma_1)(q_2,\gamma_2)\cdots$ is a *run* of $\mathcal{M}$ on α if and only if $(q_0,\gamma_0) = (q_{in},\bot)$ and for all $i \in \mathbb{N}$ exists $a_i \in \Sigma \cup \{\varepsilon\}$ such that $(q_i,\gamma_i) \overset{a_i}{\vdash} (q_{i+1},\gamma_{i+1})$ and $a_0a_1a_2\cdots = \alpha$. For a run ρ we define the set of states visited in ρ as $\mathrm{Occ}(\rho) = \{q \in Q \mid \exists j\colon \rho(j) = (q,\gamma_j) \text{ for some } \gamma_j\}$. Similarly, we define the set of states visited infinitely often in ρ as $\mathrm{Inf}(\rho) = \{q \in Q \mid \forall i \exists j > i\colon \rho(j) = (q,\gamma_j) \text{ for some } \gamma_j\}$.

A *parity pushdown automaton* (parity-DPDA) is a tuple $\mathcal{A} = (\mathcal{M}^\mathcal{A}, \mathrm{col})$ where $\mathcal{M}^\mathcal{A}$ is a DPDM and $\mathrm{col}\colon Q \to [d]$ is a priority function assigning to each state of $\mathcal{M}^\mathcal{A}$ a natural number. It accepts an ω-word $\alpha \in \Sigma^\omega$ if there exists a run ρ of $\mathcal{A}$ on α, such that $\min\{\mathrm{col}(q) \mid q \in \mathrm{Inf}(\rho)\}$ is even. The set of ω-words accepted by a parity-DPDA $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

2.2.1 Weak Automata

A *weak-parity pushdown automaton* (weak-parity-DPDA) is a tuple $\mathcal{A} = (\mathcal{M}^\mathcal{A}, \mathrm{col})$ where $\mathcal{M}^\mathcal{A}$ is a DPDM and $\mathrm{col}\colon Q \to [d]$ is a priority function assigning to each state of $\mathcal{M}^\mathcal{A}$ a natural number. It accepts an ω-word $\alpha \in \Sigma^\omega$ if there exists a run ρ of $\mathcal{A}$ on α, such that $\min\{\mathrm{col}(q) \mid q \in \mathrm{Occ}(\rho)\}$ is even. The set of ω-words accepted by a weak-parity-DPDA $\mathcal{A}$ is again denoted by $L(\mathcal{A})$.

A weak-parity-DPDA is an E-DPDA, if $\mathrm{col}(Q) \subseteq \{0,1\}$, i.e., a run is accepting if it visits a state with priority 0 at least once. A weak-parity-DPDA is an A-DPDA, if $\mathrm{col}(Q) \subseteq \{1,2\}$, i.e., a run is accepting if it never visits a state with priority 1.

2.2.2 Visibly and One-counter Automata

A DPDM is *one-counter*, if Γ is a singleton set. A *visibly pushdown alphabet* $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{int}$ is an alphabet partitioned into three disjoint alphabets: Σ_c is a set of *calls*, Σ_r a set of *returns*, Σ_{int} is a set of *internal actions*. A *deterministic visibly pushdown machine* is a DPDM $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{in}, \bot)$ where Σ is a visibly pushdown alphabet and the transition function is composed of three functions $\delta = \delta_c \cup \delta_r \cup \delta_{int}$ where $\delta_c \colon Q \times \Sigma_c \times \Gamma_\bot \to Q \times \Gamma$, $\delta_r \colon Q \times \Sigma_r \times \Gamma_\bot \to Q$, and $\delta_{int} \colon Q \times \Sigma_{int} \times \Gamma_\bot \to Q$. A deterministic visibly pushdown machine can be seen as a DPDM with transition function δ' by defining

- $\delta'(q, a, A) = (q', A'A)$, if $a \in \Sigma_c$ and $\delta_c(q, a, A) = (q', A')$,
- $\delta'(q, a, A) = \begin{cases} (q', \varepsilon) & \text{if } A \neq \bot, \\ (q', \bot) & \text{if } A = \bot, \end{cases}$ if $a \in \Sigma_r$ and $\delta_r(q, a, A) = q'$, and
- $\delta'(q, a, A) = (q', A)$, if $a \in \Sigma_{int}$ and $\delta_{int}(q, a, A) = q'$.

Note that a deterministic visibly pushdown machine treats a return symbol as an internal action when the stack is empty.

Any type of DPDA introduced above is called *visibly one-counter*, denoted by DV1CA, if the underlying DPDM is visibly and one-counter. We prefix the abbreviation DV1CA by the type of acceptance condition of the automaton (parity, weak-parity, E, or A).

3 Decision Problems

In this section, we consider various decision problems regarding delay games with context-free winning conditions. We begin by showing that the winner for a fixed delay function with bounded distance function can be determined effectively. Then, we show that determining whether Player O has a winning strategy for some finite delay is undecidable. As a corollary we obtain that determining whether Player O has a winning strategy for some constant-delay or linear-delay function is undecidable, too. We conclude by giving a general criterion to classify the sets $\mathcal{F}$ of delay functions for which it is decidable whether Player O can win a given delay game with some function from $\mathcal{F}$.

As we consider winning conditions L that are recognizable by parity-DPDA, $\Gamma_f(L)$ can be modeled as a parity game on a countable arena with finitely many priorities. Since parity games are determined [4, 10], we conclude that delay games are also determined.

▶ **Theorem 2.** *Let $\mathcal{A}$ be a parity-DPDA and $f \colon \mathbb{N} \to \mathbb{N}_+$. Then, $\Gamma_f(L(\mathcal{A}))$ is determined.*

By encoding the delay into the winning condition the following theorem is obtained. Note that the property "$\{i \mid f(i) \neq 1\}$ is finite" covers all constant-delay functions f.

▶ **Theorem 3.** *The following problem is decidable:*
Input: *Parity-DPDA $\mathcal{A}$ and $f \colon \mathbb{N} \to \mathbb{N}_+$ such that $\{i \mid f(i) \neq 1\}$ is finite.*
Question: *Does Player O win $\Gamma_f(L(\mathcal{A}))$?*

Proof. Let $\sharp$ be a letter not occurring in Σ_O. For a word $\beta = \beta(0)\beta(1)\beta(2)\cdots \in \Sigma_O^\omega$, define $\mathrm{shift}_f(\beta) = \beta'(0)\beta'(1)\beta'(2)\cdots \in (\Sigma_O \cup \{\sharp\})^\omega$ by

$$\beta'(n) = \begin{cases} \beta(i) & \text{if } n = \left(\sum_{j=0}^{i} f(j)\right) - 1 \text{ for some } i, \\ \sharp & \text{otherwise.} \end{cases}$$

Figure 1 shows an example for this operation.

β:	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	$\cdots$
$\mathrm{shift}_f(\beta)$:	$\sharp$	$\sharp$	a_0	a_1	$\sharp$	a_2	$\sharp$	a_3	a_4	a_5	$\cdots$

Figure 1 The shift_f-operation where $f(0) = 3$, $f(2) = 2$, $f(3) = 2$, and $f(i) = 1$ otherwise.

Now, let $L = L(\mathcal{A})$ and define

$$L_{\mathrm{shift}_f} = \left\{ \begin{pmatrix}\alpha(0)\\ \beta'(0)\end{pmatrix}\begin{pmatrix}\alpha(1)\\ \beta'(1)\end{pmatrix}\begin{pmatrix}\alpha(2)\\ \beta'(2)\end{pmatrix}\cdots \;\middle|\; \begin{pmatrix}\alpha(0)\\ \beta(0)\end{pmatrix}\begin{pmatrix}\alpha(1)\\ \beta(1)\end{pmatrix}\begin{pmatrix}\alpha(2)\\ \beta(2)\end{pmatrix}\cdots \in L \text{ and } \mathrm{shift}_f(\beta(0)\beta(1)\beta(2)\cdots) = \beta'(0)\beta'(1)\beta'(2)\cdots \right\} .$$

Since L is deterministic context-free and $\{i \mid f(i) \neq 1\}$ is finite, L_{shift} is also deterministic context-free. Furthermore, Player p wins $\Gamma_f(L)$ if and only if she wins the game $\Gamma_g(L_{\mathrm{shift}_f})$ where $g(i) = 1$ for all $i \in \mathbb{N}$. As $\Gamma_g(L_{\mathrm{shift}_f})$ is a game without delay, its winner can be determined effectively [12]. ◀

It turns out that Theorem 3 is the most general decidability result for arbitrary parity-DPDA: relaxing the finiteness condition on $\{i \mid f(i) \neq 1\}$ makes the problem undecidable (see Theorem 6).

We continue with the undecidability results which are obtained by a reduction from the halting problem for 2-register machines. A 2-register machine $\mathcal{R}$ is a list $(0\colon I_0), \ldots, (k-2\colon I_{k-2}), (k-1\colon \texttt{STOP})$, where the first entry of a pair $(\ell\colon I_\ell)$ is the line number and the second one is the instruction, which is of the form $\texttt{INC(X}_i\texttt{)}$, $\texttt{DEC(X}_i\texttt{)}$, or $\texttt{IF X}_i\texttt{=0 GOTO m}$ where $i \in \{0, 1\}$ is the number of a register and $m \in [k]$. A configuration of $\mathcal{R}$ is a tuple (ℓ, n_0, n_1) where $\ell \in [k]$ is a line number and $n_0, n_1 \in \mathbb{N}$ are the contents of the registers. The semantics are defined in the obvious way with the convention that a decrease of a register holding a zero has no effect. We say that $\mathcal{R}$ halts, if it reaches a configuration $(k-1, n_0, n_1)$ for some $n_0, n_1 \in \mathbb{N}$ when started with the initial configuration $(0, 0, 0)$. It is well-known that the halting problem for 2-register machines is undecidable [11].

▶ **Theorem 4.** *The following problem is undecidable:*
Input: *Parity-DPDA $\mathcal{A}$.*
Question: *Is there a delay function f such that Player O wins $\Gamma_f(L(\mathcal{A}))$?*

Proof. We proceed by a reduction from the halting problem for 2-register machines. Given such a machine $\mathcal{R} = (0\colon I_0), \ldots, (k-2\colon I_{k-2}), (k-1\colon \texttt{STOP})$, we encode a configuration (ℓ, n_0, n_1) by the word $\ell r_0^{n_0} r_1^{n_1}$. Note that if c encodes a configuration, then we have $|c'| \leq |c| + 1$ for the encoding c' of the successor configuration.

Now, define $\mathrm{Conf} = \{\ell r_0^{n_0} r_1^{n_1} \mid \ell \in [k],\ n_0, n_1 \in \mathbb{N}\}$, $\mathrm{Conf}_0 = 0$, and consider the following game specification over $\Sigma_I \times \Sigma_O$, where $\Sigma_I = \{\sharp, r_0, r_1\} \cup [k]$ and $\Sigma_O = \{N, E_0, E_1, L\}$: Player I builds up a word of the form $\sharp\mathrm{Conf}_0(\sharp\mathrm{Conf})^\omega$ (if he does not, he loses). Consider such a word $\sharp c_0 \sharp c_1 \sharp c_2 \sharp \cdots$ with $c_0 = \mathrm{Conf}_0$ and $c_i \in \mathrm{Conf}$ for all $i > 0$. In order to win, Player O has to find a pair c_j, c_{j+1} such that c_{j+1} does not encode the successor configuration of the configuration encoded by c_j. To do this, she indicates at each position where Player I has played a $\sharp$ whether she believes that the following two configurations are indeed successive configurations (by playing the letter N) or whether she claims an error (by playing E_0, E_1, L indicating that the first register, the second register, or the line number is not updated correctly). At any other position, she may pick an arbitrary letter.

$$\cdots\ \begin{matrix}\sharp\\ N\end{matrix}\ \overbrace{\begin{matrix}3 & r_0 & r_0 & r_0 & r_1\\ * & * & * & * & *\end{matrix}}^{(3,3,1)}\ \begin{matrix}\sharp\\ E_0\end{matrix}\ \overbrace{\begin{matrix}4 & r_0 & r_0 & r_0 & r_0 & r_1\\ * & * & * & * & * & *\end{matrix}}^{(4,4,1)}\ \begin{matrix}\sharp\\ N\end{matrix}\ \overbrace{\begin{matrix}5 & r_0 & r_0 & r_0 & r_0 & r_0\\ * & * & * & * & * & *\end{matrix}}^{(5,5,0)}\ \begin{matrix}\sharp\\ N\end{matrix}\ \cdots$$

Figure 2 Part of a play encoding three configurations.

Figure 2 depicts the encoding of three configurations (here, $*$ denotes an arbitrary letter). Assuming that line 3 contains `INC(X`$_0$`)` and line 4 contains `DEC(X`$_1$`)`, then the first update is correct, while the second one is not: the first register is increased incorrectly, an error which is claimed by the letter E_0 in front of the second encoding.

This winning condition can be recognized by a parity-DPDA $\mathcal{A}_{\mathcal{R}}$. The automaton checks whether the first component is a word in $\sharp\mathrm{Conf}_0(\sharp\mathrm{Conf})^\omega$. If it encounters a letter $\binom{\sharp}{E_0}$, $\binom{\sharp}{E_1}$, or $\binom{\sharp}{L}$ it has to check the next two encodings $\ell r_0^{n_0} r_1^{n_1} \sharp \ell' r_0^{n_0'} r_1^{n_1'} \sharp$.

- Case $\binom{\sharp}{E_i}$ for $i \in \{0,1\}$: $\mathcal{A}_{\mathcal{R}}$ has to verify $n_i + s \neq n_i'$, where $s = 1$, if $I_\ell = \mathtt{INC(X}_i\mathtt{)}$, $s = -1$, if $I_\ell = \mathtt{DEC(X}_i\mathtt{)}$ and $n_i > 0$, and $s = 0$ otherwise.
- Case $\binom{\sharp}{L}$: $\mathcal{A}_{\mathcal{R}}$ has to verify $\ell + 1 \neq \ell'$, if $I_\ell = \mathtt{INC(X}_i\mathtt{)}$, $I_\ell = \mathtt{DEC(X}_i\mathtt{)}$, or $I_\ell = \mathtt{IF\ X}_i\mathtt{=0\ GOTO\ m}$ and $n_i > 0$; and $\mathcal{A}_{\mathcal{R}}$ has to verify $\ell' \neq m$, if $I_\ell = \mathtt{IF\ X}_i\mathtt{=0\ GOTO\ m}$ and $n_i = 0$.

All these tests can be implemented in terms of a parity-DPDA $\mathcal{A}_{\mathcal{R}}$ that accepts a word if and only if the first component is not a word in $\sharp\mathrm{Conf}_0(\sharp\mathrm{Conf})^\omega$ or if the first occurrence of a letter E_0, E_1, or L in the second component correctly claims an error.

We show that $\mathcal{R}$ halts if and only if there exists a delay function f such that Player O wins the game $\Gamma_f(L(\mathcal{A}_{\mathcal{R}}))$.

Suppose $\mathcal{R}$ halts and consider the linear-delay function $f(i) = 6$ for all i. We claim that Player O has a winning strategy for $\Gamma_f(L(\mathcal{A}_{\mathcal{R}}))$ which finds the first error introduced by Player I. In round 0 Player I chooses 6 letters which are sufficient for Player O to check whether Player I has encoded the initial configuration and its successor configuration, as the length of such an encoding is bounded by 6. Now consider a round $i > 0$: if the i-th input letter is not a $\sharp$, then Player O can choose an arbitrary output letter. So suppose that it is a $\sharp$ and that Player O has not yet signaled an error up to this position: Player I has produced a word $\sharp x \sharp y$ of length $6(i+1)$ where $|x| = i - 1$ and hence, $|y| = 5(i+1)$. Note that both x and y might contain the letter $\sharp$. Let c denote the last encoding of a configuration in x and c' the first encoding of a configuration in y. As Player O has not signaled an error at the previous $\sharp$, we know that c' is well-defined and that it is the encoding of the successor configuration of the configuration encoded by c. We have $|c| \leq |x| = i - 1$ and hence $|c'| \leq i$. Thus, the successor configuration of c' is encoded by at most $i + 1$ letters. As $i + (i+1) + 2 < 5(i+1)$ for all $i > 0$, in every round Player O has enough information to detect an error if one is introduced. This strategy is indeed winning for Player O as an error will eventually be introduced, since a halting configuration has no successor.

Now suppose $\mathcal{R}$ does not halt. Player I has a winning strategy in $\Gamma_f(L(\mathcal{A}_{\mathcal{R}}))$ for any function f by building up the word $\sharp c_0 \sharp c_1 \sharp c_2 \sharp \cdots$ where $c_0, c_1, c_2, \ldots$ are the encodings of the infinite run of $\mathcal{R}$ starting with the initial configuration. Hence, due to determinacy, Player O does not win $\Gamma_f(L(\mathcal{A}_{\mathcal{R}}))$. ◀

The game induced by $L(\mathcal{A}_{\mathcal{R}})$ can be won by Player O with some suitable constant-delay function or with the linear-delay function with delay 6 if and only if $\mathcal{R}$ halts. Hence, we obtain the following corollary.

▶ **Corollary 5.** *The following problems are undecidable:*

1. ***Input:** Parity-DPDA $\mathcal{A}$.*
   ***Question:** Is there a constant-delay function f such that Player O wins $\Gamma_f(L(\mathcal{A}))$?*
2. ***Input:** Parity-DPDA $\mathcal{A}$.*
   ***Question:** Is there a linear-delay function f such that Player O wins $\Gamma_f(L(\mathcal{A}))$?*
3. ***Input:** Parity-DPDA $\mathcal{A}$ and $k \in \mathbb{N}$.*
   ***Question:** Let $f(i) = k$ for all i. Does Player O win $\Gamma_f(L(\mathcal{A}))$?*

By slightly modifying the game described above, one shows that all undecidability results hold even for winning conditions given by E-DV1CA. To this end, we supply Player O with additional letters C, R, Int and define $\Sigma_c = \Sigma_I \times \{C\}$, $\Sigma_r = \Sigma_I \times \{R\}$, and $\Sigma_{int} = \Sigma_I \times \{\mathit{Int}, N, L, E_0, E_1\}$, i.e., Player O controls the behavior of the stack. As soon as she has answered a $\sharp$ by one of the letters E_0, E_1, L she has to use the letters C, R, Int to enable the automaton to compare the respective parts of the following two encodings. If she fails to do so, she loses immediately. This can be implemented by means of a finite automaton, which can be combined with the parity-DPDA $\mathcal{A}_\mathcal{R}$ to obtain a visibly parity-DPDA. Furthermore, all necessary tests can be implemented using a single stack symbol. Finally, by modifying the game specification such that the first component is no longer required to be of the form $\sharp\mathrm{Conf}_0(\sharp\mathrm{Conf})^\omega$, it can be recognized by an E-DV1CA. If Player I does not produce an infinite sequence of encodings of configurations, Player O has the possibility to claim an error and thereby win.

Based on the proofs of the previous theorems, we conclude this section by giving a general criterion to determine for a set $\mathcal{F}$ of delay functions whether it is decidable whether Player O wins a given delay game with some delay function from $\mathcal{F}$. We say that a set $\mathcal{F}$ of delay functions $f\colon \mathbb{N} \to \mathbb{N}_+$ is *bounded*, if there exists a $d \in \mathbb{N}$ such that for every $f \in \mathcal{F}$ and every $i \in \mathbb{N}$ we have $d_f(i) \leq d$, i.e., there is a global bound on the lookahead for Player O given by the functions in $\mathcal{F}$. Notice that since $\mathcal{F}$ is not part of the input, we can state the following theorem for any set of delay functions without having to represent the set effectively.

▶ **Theorem 6.** *Let $\mathcal{F}$ be a set of delay functions. The following problem is decidable if and only if $\mathcal{F}$ is bounded:*

***Input:** Parity-DPDA $\mathcal{A}$.*
***Question:** Does there exist an $f \in \mathcal{F}$ such that Player O wins $\Gamma_f(L(\mathcal{A}))$?*

Proof. Consider a bounded set $\mathcal{F}$ of delay functions. We define a partial order on delay functions as follows: $f \leq g$ if and only if $d_f(i) \leq d_g(i)$ for all i, i.e., g allows at any round at least as much lookahead as f does. Applying Dickson's Lemma [3] and the boundedness of $\mathcal{F}$ one shows that there exists a finite set of maximal elements $\mathcal{F}_{max} \subseteq \mathcal{F}$ (a function $f \in \mathcal{F}$ is maximal if for all $g \in \mathcal{F}$, $f \leq g$ implies $f = g$). We claim that there exists an $f \in \mathcal{F}$ such that Player O wins $\Gamma_f(L(\mathcal{A}))$ if and only if there exists an $g \in \mathcal{F}_{max}$ such that Player O wins $\Gamma_g(L(\mathcal{A}))$. As $\mathcal{F}_{max}$ is finite and every $g \in \mathcal{F}_{max}$ satisfies "$\{i \mid g(i) \neq 1\}$ is finite", the latter property can be decided by Theorem 3.

The implication from right to left is trivially true, so assume there exists an $f \in \mathcal{F} \setminus \mathcal{F}_{max}$ such that Player O wins $\Gamma_f(L(\mathcal{A}))$. Then, there is a function $g \in \mathcal{F}_{max}$ such that $f \leq g$, i.e., the function g admits Player O at least as much lookahead as f. Hence, a winning strategy for Player O in $\Gamma_f(L(\mathcal{A}))$ can easily be turned into a winning strategy for her in $\Gamma_g(L(\mathcal{A}))$.

Now consider an unbounded set $\mathcal{F}$ of delay functions, i.e., for every $d \in \mathbb{N}$ there exists an $f \in \mathcal{F}$ and an $i \in \mathbb{N}$ such that $d_f(i) > d$. We adapt the specification described in the proof of Theorem 4 by allowing Player O to postpone the beginning of the simulation of a computation of $\mathcal{R}$ until she has attained enough lookahead to inspect the complete halting computation of $\mathcal{R}$ (if there exists one) before she has to indicate potential errors.

Given a 2-register machine $\mathcal{R}$ with k instructions, define Conf_0 and Conf as in the proof of Theorem 4, and consider the following game specification over $\Sigma_I \times \Sigma_O$, where $\Sigma_I = \{\sharp, r_0, r_1, \$\} \cup [k]$ and $\Sigma_O = \{N, E_0, E_1, L, S, \$\}$: Player I builds a word of the form $\$^*\mathrm{Conf}_0(\sharp\mathrm{Conf})^\omega$ or $\$^\omega$. If he does not adhere to the format, he loses. Player I may produce the word $\$^\omega$ if and only if Player O never plays the letter S to start the simulation. If Player O plays the letter S, then Player I has to play a word of the form $\$^*c_0\sharp c_1\sharp c_2\sharp\cdots$ with $c_0 = \mathrm{Conf}_0$ and $c_i \in \mathrm{Conf}$ for every $i > 0$. Again, in order to win, Player O has to find a pair c_j, c_{j+1} such that c_{j+1} does not encode the successor configuration of the configuration encoded by c_j. The mechanism to do so is similar to the one described in the proof of Theorem 4. We denote the parity-DPDA recognizing this winning condition by $\mathcal{A}'_{\mathcal{R}}$.

Suppose $\mathcal{R}$ halts after n computation steps. Then, the full computation of $\mathcal{R}$ is encoded by at most $d = \sum_{j=1}^{n+1}(j+1)$ letters. Let $f \in \mathcal{F}$ and $i \in \mathbb{N}$ such that $d_f(i) \geq d$. Player O has a winning strategy in $\Gamma_f(L(\mathcal{A}'_{\mathcal{R}}))$. In the first i rounds, she chooses \$. If Player I has picked in a round $j \leq i+1$ a word $u_j \neq \$^{f(j)}$, then Player O wins by playing \$ ad infinitum. Otherwise, she plays S in round $i+1$. Hence, in order to win Player I has to start simulating $\mathcal{R}$, say at position $j > i$. As d_f is non-decreasing, Player O has at least d letters lookahead when picking her letter in any round $j' \geq j$. As the machine halts, this lookahead enables her to detect an error which Player I has to introduce, since a halting configuration does not have a successor configuration.

If $\mathcal{R}$ does not halt, then Player I has a winning strategy in $\Gamma_f(L(\mathcal{A}'_{\mathcal{R}}))$ for every delay function $f \in \mathcal{F}$: as long as Player O has not played S, pick $\$^{f(i)}$ in round i. As soon as she has played S, he starts producing the word $\sharp c_0\sharp c_1\sharp c_2\sharp\cdots$, where $c_0, c_1, c_2, \ldots$ are encodings of the infinite run of $\mathcal{R}$ starting in the initial configuration. ◀

Using the ideas presented above, one can show that Theorem 6 holds even for weak-parity-DV1CA. However, E-acceptance and A-acceptance are not sufficient, since Player O has to be forced to play an S and Player I has to be forced to start the simulation after Player O played an S.

4 Lower Bounds on Delays

In this section we show that there exists a deterministic context-free winning condition L such that Player O wins the game $\Gamma_f(L)$ for some delay function f, but not for any elementary-delay function. To this end, we adapt the idea of the previous section: Player I produces an ω-word which can be decomposed into blocks on which a successor relation is defined. In order to win, Player O has to find a pair of consecutive blocks that are not in the successor relation and the game specification forces Player I to produce such an error at some point. In contrast to the specifications of the previous section, Player O does not signal a potential error in front of the i-th block, but with her i-th bit. By ensuring that a valid successor block is exponentially longer than its predecessor we obtain our result.

▶ **Theorem 7.** *There exists a parity-DPDA $\mathcal{A}$ and a delay function f such that Player O wins $\Gamma_f(L(\mathcal{A}))$, but Player I wins $\Gamma_{f'}(L(\mathcal{A}))$ for every elementary-delay function f'.*

Proof. Let $S_\sharp = \{\sharp_N, \sharp_D, \sharp_C\}$ and $S_\flat = \{\flat_N, \flat_H\}$ be two sets of signals for Player I and define $B = S_\flat 0(S_\flat 0^+)^*$ and $B_0 = S_\flat 0$. We say that a *block* $w = \flat_0 0 \flat_1 0^{n_1} \flat_2 \cdots \flat_{k-1} 0^{n_{k-1}} \in B$ has k $\flat$*-blocks*. Consider a word $\sharp_0 w_0 \sharp_1 w_1 \sharp_2 w_2 \sharp_3 \cdots$ where $w_0 \in B_0$ and $w_i \in B$, $\sharp_i \in S_\sharp$ for all i. We say that a block $w_i = \flat_0 0^{n_0} \flat_1 0^{n_1} \flat_2 \cdots \flat_{k-1} 0^{n_{k-1}}$ has a *doubling error* at position j in the

$$\begin{array}{c}\sharp_N\\ N\end{array}\overbrace{\begin{array}{cc}\flat_N & 0\\ E_D & N\end{array}}^{w_0}\begin{array}{c}\sharp_C\\ N\end{array}\overbrace{\begin{array}{cccc}\flat_N & 0 & \flat_N & 0\\ H & N & N & N\end{array}}^{w_1}\begin{array}{c}\sharp_N\\ N\end{array}\overbrace{\begin{array}{cccccccccc}\flat_N & 0 & \flat_N & 0 & 0 & \flat_N & 0 & 0 & 0 & 0\\ N & N & N & N & N & N & N & N & N & N\end{array}}^{w_2}\begin{array}{c}\sharp_N\\ N\end{array}\ \cdots$$

Figure 3 A play prefix with three blocks w_0, w_1, and w_2.

range $0 \leq j < k-1$ if $n_{j+1} \neq 2n_j$ (note that $n_0 = 1$ for every $w_i \in B$). The doubling error at position j in block w_i is *signaled*, if $\sharp_i = \sharp_D$, $\flat_j = \flat_H$, and $\flat_{j'} = \flat_N$ for all $j' < j$. We say that two consecutive blocks w_i and w_{i+1} constitute a *copy error*, if $|w_i| \neq |w_{i+1}|_{\flat_N} + |w_{i+1}|_{\flat_H}$ (i.e., in the absence of a copy error, w_{i+1} has $|w_i|$ $\flat$-blocks). For two blocks w_i and w_{i+1} the copy error is *signaled* if $\sharp_i = \sharp_C$.

Figure 3 depicts the encoding of three blocks in the first component. The blocks w_1 and w_2 constitute a copy error, as w_2 contains only three $\flat$-blocks, and not the required four (due to $|w_1| = 4$). This error is signaled by the letter $\sharp_C$ in front of w_1. Furthermore, the block w_1 contains a doubling error at position 0 which is not signaled.

Consider the following game specification over $\Sigma_I \times \Sigma_O$, where $\Sigma_I = \{0\} \cup S_\sharp \cup S_\flat$ and $\Sigma_O = \{N, E_D, E_C, H\}$: Player I builds a word $\alpha = \sharp_0 w_0 \sharp_1 w_1 \sharp_2 w_2 \sharp_3 \cdots \in S_\sharp B_0 (S_\sharp B)^\omega$ while Player O produces a word $\beta \in \Sigma_O^\omega$. Player O uses her letters to announce errors in α: if $\beta(i) = E_C$, then she claims that the pair w_i and w_{i+1} constitutes a copy error. If $\beta(i) = E_D$, then she claims that $w_i = \flat_0 0^{n_0} \flat_1 0^{n_1} \flat_2 \cdots \flat_{k-1} 0^{n_{k-1}}$ contains a doubling error at a position j, which she has to specify by answering $\flat_j$ by H (and every $\flat_{j'}$ for $j' < j$ not by H).

Going back to the example in Figure 3, we see that Player O claims the doubling error in w_1 by choosing $\beta(1) = E_D$ and identifying its position by playing H directly in front if it.

Player O wins a play if and only if the induced word $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ satisfies at least one of the following conditions.

- $\alpha \notin S_\sharp B_0 (S_\sharp B)^\omega$ (i.e., Player I does not adhere to the format).
- There exists an i such that $\beta(i) = E_D$, $\beta(j) = N$ for all $j < i$, $\sharp_j = \sharp_N$ for all $j < i$, the ℓ-th $\flat$ of w_i is answered by H (and ℓ is minimal with this property) and w_i contains a doubling error at position ℓ (i.e., Player O detects a doubling error in block w_i and there is no error signaled by Player I in front of any block w_j for some $j < i$. Note that the doubling error in block w_i may have been signaled by Player I).
- There exists an i such that $\beta(i) = E_C$, $\beta(j) = N$ for all $j < i$, $\sharp_j = \sharp_N$ for all $j < i$, and the pair w_i and w_{i+1} constitutes a copy error (i.e., Player O detects a copy error in blocks w_i and w_{i+1} and there is no error signaled by Player I in front of any block w_j for some $j < i$. Note that the copy error in the blocks w_i and w_{i+1} may have been signaled by Player I in front of w_i).
- There exists an i such that $\sharp_i = \sharp_D$ and $\sharp_j = \sharp_N$ for all $j < i$, and $\beta(j) = N$ for all $j \leq i$, and w_i does not contain $\flat_H$ or the two blocks following the first $\flat_H$ do not constitute a doubling error (i.e., Player I signals a doubling error but does not indicate its position correctly).
- There exists an i such that $\sharp_i = \sharp_C$ and $\sharp_j = \sharp_N$ for all $j < i$, and $\beta(j) = N$ for all $j \leq i$, and the pair w_i and w_{i+1} does not constitute a copy error (i.e., Player I signals a copy error without producing one).
- $\sharp_i = \sharp_N$ for all i (i.e., Player I never signals an error).

Hence, the play indicated in Figure 3 is winning for Player O as her correct claim $\beta(1) = E_D$ precedes the signal $\sharp_1 = \sharp_C$ of Player I.

Let $L = \{\rho \in (\Sigma_I \times \Sigma_O)^\omega \mid \rho \text{ is winning for Player } O\}$. We show that L can be recognized by a parity-DPDA $\mathcal{A}$: on an ω-word $\rho = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ where $\alpha = \sharp_0 w_0 \sharp_1 w_1 \sharp_2 w_2 \sharp_3 \cdots \in S_\sharp B_0 (S_\sharp B)^\omega$, $\mathcal{A}$ proceeds in four phases described below. If α is not of the required format or contains no letter $\sharp_C$ or $\sharp_D$, then ρ is accepted.

In the first phase, it prepares its stack to be able to find the beginning of w_i when starting at letter $\rho(i)$ as required in the second phase. To do so, it counts the number of letters processed so far minus the number of letters from $S_\sharp$ in the first component. This phase is stopped as soon as a letter $\sharp_C$ or $\sharp_D$ in the first component is read or a letter E_C or E_D in the second component is read. In the first case, the automaton jumps to phase four, in the second it starts with phase two.

The second phase starts if $\beta(i)$ is E_C or E_D for the first time. Then, the automaton uses the information on the stack to find the beginning of w_i by decreasing the stack every time a $\sharp_N$ in the first component is processed. If $\sharp_j = \sharp_C$ or $\sharp_j = \sharp_D$ for $j < i$ is processed, the automaton jumps to phase four. Otherwise, phase two continues until the beginning of w_i is reached. Then, $\mathcal{A}$ continues with phase three.

In phase three, $\mathcal{A}$ checks whether the error indicated by $\beta(i)$ occurs (for this purpose, it stores $\beta(i)$ at the beginning of phase two). If $\beta(i) = E_C$, then it checks whether w_i and w_{i+1} constitute a copy error. If $\beta(i) = E_D$, then it checks whether w_i contains a doubling error, which has to be indicated in the second component by an H right before the error. If the first H does not indicate an error correctly (or if none is read), then $\mathcal{A}$ rejects ρ. The automaton accepts in phase three if and only if the error indicated by $\beta(i)$ is detected.

Finally, in phase four $\mathcal{A}$ checks whether the error indicated by $\sharp_j$ occurs. If $\sharp_j = \sharp_C$, then it checks whether w_j and w_{j+1} constitute a copy error. If $\sharp_j = \sharp_D$, then it checks whether w_j contains a doubling error, which is signaled properly by a $\flat_H$ at the appropriate position. If the first $\flat_H$ does not indicate an error correctly (or if w_j does not contain a $\flat_H$), then $\mathcal{A}$ accepts ρ. The automaton accepts in phase four if and only if the error indicated by $\sharp_j$ is not detected.

All the tests described in phases three and four can be implemented by a parity-DPDA.

We continue by showing that there exists a delay function f such that Player O wins the game $\Gamma_f(L)$. To this end, note that the following holds true for two consecutive blocks w and w' not containing a copy or doubling error: $|w'| = 2^{|w|} + |w| - 1$. Hence, we define the auxiliary function g by $g(0) = 2$ and $g(n+1) = 2^{g(n)} + g(n) - 1$ for every $n \geq 0$. Now, define the delay function f by $f(0) = g(0) + g(1) + 3$ and $f(n) = g(n+1) + 1$ for every $n > 0$ (note that f is non-elementary). We claim that Player O has a winning strategy for $\Gamma_f(L)$: if Player I does not pick $\sharp_0 \flat_{00} 0 \sharp_1 \flat_{10} 0 \flat_{11} 00 \sharp_2$ in the first round, then he has committed some error within his first two blocks, which can be claimed by Player O with v_0. Now assume he has produced a play prefix $\sharp_0 w_0 \sharp_1 w_1 \sharp_2 \cdots \sharp_i w_i \sharp_{i+1}$ after round $i-1$ without introducing a doubling error in the blocks w_j for all $j < i$ and no copy error in the pairs w_j and w_{j+1} for all $j < i$. If he produces an x in the next round i that is of the form $w\sharp$ such that w_i and w do not constitute a copy error and if w_i does not contain a doubling error, then Player O picks $v_i = N$. Otherwise, she claims the error that occurs. This strategy is winning for Player O, as Player I is not able to signal and produce an error that cannot be claimed by Player O.

Finally, consider an elementary-delay function $f_e \in \mathcal{O}(\exp_k)$. Player I can always play blocks without introducing errors until the length of the block w_i exceeds the lookahead $\left(\sum_{j=0}^{i} f_e(j)\right) - i$ of Player O. At such a position, Player O has to make a claim concerning a block which is not completed yet. So, Player I signals a doubling error for this incomplete block. If Player O does not claim a doubling error, then he can introduce a doubling error while completing the block. Then, Player I wins, if he sticks to the input format, since he is the first to claim an error. Vice versa, if Player O claims the doubling error, then Player I does not introduce a doubling error while completing the block. Then he continues to stick to the input format and wins, as his claim is preceded by the claim of Player O. ◀

Using ideas as presented in Section 3 one can show that Theorem 7 holds even for A-DV1CA. However, the game specification as described above, cannot be accepted by a visibly one-counter automaton: the problem arises if the automaton has to change from phase two to phase four. In this situation, the stack is not yet empty and the automaton has to check a claimed error. To do this using one stack symbol, the stack has to be emptied before the next letter is processed, which cannot be done by a visibly automaton, as it has no ε-transitions. To resolve this, we modify the game specification such that if this situation occurs (changing from phase two to phase four), Player O loses immediately. Player O has still the possibility to win by additionally never claiming an error in a block w_i if Player I already claimed an error in a block w_j for some $j < i$. This modified game specification is visibly one-counter. Finally, as a play is only winning for Player I, if he claims an existing error before Player O does, the set of winning plays for Player O can be accepted by an A-DV1CA.

5 Conclusion

In this paper we continued the investigation of delay games. We showed that determining the winner of deterministic context-free delay games is undecidable. Also, we presented a game that is won by Player O with finite delay, but the necessary lookahead is non-elementary. Both results already hold for the restricted class of winning conditions recognized by visibly one-counter automata with weak acceptance conditions.

Our undecidability results and lower bounds on the delay for visibly winning conditions hold even for the more restricted case where Player O controls the behavior of the stack (more formally, the membership of a letter to Σ_c, Σ_r, or Σ_{int} respectively, depends only on its second component). An interesting open question is whether these results also hold if Player I obtains control over the stack behavior, i.e., the first component of a letter determines to which alphabet it belongs. The following example shows that linear delay is necessary in this case, even for one-counter winning conditions.

Let $\Sigma_I = \{c, r\}$ and $\Sigma_O = \{0, 1\}$. We partition the alphabet $\Sigma_I \times \Sigma_O$ into the alphabets $\Sigma_c = \{c\} \times \Sigma_O$ and $\Sigma_r = \{r\} \times \Sigma_O$, i.e., Player I controls the behavior of the stack. Consider the following game specification L with $\rho = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots \in L$ if one of the following conditions holds:

- $\alpha(0) = r$, i.e., Player O wins immediately if Player I's first letter is a return,
- $|\alpha(0)\cdots\alpha(n)|_c > |\alpha(0)\cdots\alpha(n)|_r$ for all $n \in \mathbb{N}$, i.e., the stack is never empty, or
- for the minimal n such that $|\alpha(0)\cdots\alpha(n)|_c = |\alpha(0)\cdots\alpha(n)|_r$ we have $\beta(m) = 1$ and $\beta(m') = 0$ for all $m' < m$, where $m < n$ is the maximal position such that $\alpha(m) = c$, i.e., Player O indicates the last call position before the stack is empty for the first time.

This winning condition requires a linear-delay function with delay at least 2 for Player O, which is also sufficient for her to win. This is due to the fact that the stack height after processing n letters is bounded by n. Hence, Player I can play at most n returns before the stack is empty.

It is open whether linear delay is always sufficient for visibly winning conditions, if Player I controls the behavior of the stack. Moreover, it is open whether the winner of such a game can be determined effectively.

Acknowledgments

We want to thank Michael Holtmann for helpful discussions.

References

1 Alur, R., Madhusudan, P.: Adding nesting structure to words. J. ACM, 56(3) (2009)

2 Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Trans. Amer. Math. Soc. 138, 295–311 (1969)

3 Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Amer. J. Math. 35(4), 413–422 (1913)

4 Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: FOCS 91. IEEE (1991)

5 Finkel, O.: Topological properties of omega context-free languages. Theor. Comput. Sci., 262(1), 669–697 (2001)

6 Grädel, E., Thomas, W., Wilke, T. (eds): Automata, logics, and infinite Games. LNCS, vol. 2500, Springer, Heidelberg (2002)

7 Holtmann, M., Kaiser, Ł., Thomas, W.: Degrees of lookahead in regular infinite games. In: Ong, L., (ed), FOSSACS 2010. LNCS, vol. 6014, pp. 252–266, Springer, Heidelberg (2010)

8 Hosch, F., Landweber, L.H.: Finite delay solutions for sequential conditions. In: Nivat, M. (ed.), ICALP 1972, pp. 45–60. North-Holland, Amsterdam (1972)

9 Löding, C., Madhusudan, P., Serre, O.: Visibly pushdown games. In: Lodaya, K., Mahajan, M., (eds), FSTTCS 2004. LNCS, vol. 3328, pp. 408–420, Springer, Heidelberg (2004)

10 Mostowski, A.W.: Games with forbidden positions. Technical report 78, University of Gdańsk, Poland (1991)

11 Shepherdson, J. C., Sturgis, H. E.: Computability of recursive functions. J. ACM, 10(2), 217–255 (1963)

12 Walukiewicz, I.: Pushdown processes: games and model-checking. Inf. Comput., 164(2), 234–263 (2001)