RWTH Aachen

Department of Computer Science
Technical Report

Degrees of Lookahead in Context-free Infinite Games

Wladimir Fridman, Christof Löding, and Martin Zimmermann

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2010-20

RWTH Aachen · Department of Computer Science · December 2010

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

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 opponents moves. We show that the problem of determining the winner of such a game is undecidable for context-free winning conditions. Furthermore, we show that the necessary lookahead to win a context-free delay game cannot be bounded by an elementary function.

1 Introduction

Many of todays 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 (moves in alternation or concurrent), zero-sum or nonzero-sum, 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 effectively [10]. On the other hand, Finkel showed that the problem of determining the winner of a game with (non-deterministic) context-free winning condition is undecidable [5]. Hence, in the following, when we write context-free we mean deterministic context-free. In [7, 8], infinite games with delay are considered. In such a 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 delay. On the other hand, there are simple winning conditions that cannot be won with any finite delay. Hosch and Landweber proved that a delay game with regular winning condition is won by the player with lookahead if and only if she can win with (triply-exponential) constant delay, i.e., the lookahead which is necessary to win is bounded [8]. This result was improved by Holtmann, Kaiser, and Thomas [7] who showed that doubly-exponential constant delay suffices and gave a streamlined proof of this

* Supported by the project *Games for Analysis and Synthesis of Interactive Computational Systems (GASICS) of the European Science Foundation.*

result. Also, they mention that constant delay does not always suffice to win a delay game with a context-free winning condition. They ask whether the winner of a delay game with context-free winning condition can be determined effectively and what kind of delay functions are necessary to win.

We answer these questions for several classes of context-free winning conditions. Firstly, by slightly extending the technique of Walukiewicz [10] it is easy to see that determining the winner for a fixed constant delay function (i.e., there is a bound d such that the lookahead is always less than d) is decidable. Then, we show that determining whether there exists a delay function such that a given player wins the game with this delay function is undecidable for context-free winning conditions. We complement this by giving a criterion to determine when this question is decidable if we restrict the set of possible delay functions. Intuitively, if there is no global bound on the lookahead attained by all functions in the given set, then determining the winner is undecidable. If there is such a bound, then it follows from the positive result mentioned above 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 pushdown automata [1]. 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, pop, or skip (the stack is not changed) transition. Visibly pushdown automata are a popular choice to model recursive processes as their languages are closed under all boolean operations (as opposed to context-free languages in general), they can be determinized, and have good algorithmic properties (for a thorough discussion see [1]).

Finally, we consider the delay necessary to win delay games with context-free winning conditions. We present a winning condition that can be won with finite delay, but not with any delay function 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 one-counter automaton and by a visibly pushdown automaton. However, it is not clear whether a similar result holds for visibly one-counter pushdown automata.

Our results show that, unlike in the regular case, adding delay to context-free games, significantly changes their algorithmic properties. Constant delay, which is sufficient to win regular games, can always be encoded into the winning condition, hence the classical algorithms to solve games without delay are still applicable. However, in case of context-free games, where unbounded lookahead is necessary, one cannot encode the delay 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 to specify winning conditions. Then, in Section 3 we present the decidability and undecidability results. The lower bound for the delay is presented in Section 4. We conclude with some open questions in Section 5.

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 $\{0, \cdots, n-1\}$. 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.

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 $i = 0, 1, 2, \ldots$ 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 $(u_0, v_0), (u_1, v_1), (u_2, v_2), \ldots$ as a *play* of $\Gamma_f(L)$, which gives 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 $(u_0, v_0), (u_1, v_1), (u_2, v_2), \ldots$ 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.

For a delay function $f \colon \mathbb{N} \to \mathbb{N}_+$ define its *distance function* d_f by $d_f(i) = \sum_{j=0}^{i} f(j) - (i+1)$. 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 that f is an *elementary delay function*, if $d_f \in \mathcal{O}(\exp_k)$ for some fixed k. Given an ω-language L, we say that Player p *wins the game induced by L with constant, linear, elementary,* or *finite delay*, if there exists a constant, linear, elementary, or arbitrary delay function $f \colon \mathbb{N} \to \mathbb{N}_+$ such that Player p has a winning strategy for $\Gamma_f(L)$.

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

A stack content is a word from $\Gamma^* \perp$, 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 $\gamma \in \Gamma^* \perp$. We write $(q, A\gamma) \overset{a}{\vdash} (q', \gamma'\gamma)$, if $(q', \gamma') = \delta(q, a, A)$ for $a \in \Sigma \cup \{\varepsilon\}$, $\gamma, \gamma' \in \Gamma_\perp^*$ and $A \in \Gamma_\perp$.

For an ω-word $\alpha = \alpha(0)\alpha(1)\alpha(2) \cdots \in \Sigma^\omega$ an infinite sequence of configurations $\rho = (q_0, \gamma_0)(q_1, \gamma_1)(q_2, \gamma_2) \cdots$ is an *run* of $\mathcal{M}$ on α if and only if $(q_0, \gamma_0) = (q_{in}, \perp)$ 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 $a_0a_1a_2\cdots = \alpha$. We define the set of states seen infinitely often in a run ρ as $\mathrm{Inf}(\rho) = \{q \in Q \mid \forall i \in \mathbb{N}\ \exists j > i \colon \rho(j) = (q, \gamma_j) \text{ for some } \gamma_j\}$.

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 $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

One-counter and visibly automata. A DPDM is a *one-counter DPDM*, 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 $\delta_c \colon Q \times \Sigma_c \times \Gamma_\bot \to Q \times \Gamma$, $\delta_r \colon Q \times \Sigma_r \times \Gamma \to Q$ 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 setting $\delta'(q, a, A) = (q', A'A)$, if $a \in \Sigma_c$ and $\delta_c(q, a, A) = (q', A')$, $\delta'(q, a, A) = (q', \varepsilon)$, 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 cannot process a letter from Σ_r, if its stack is empty.

A parity-DPDA is called *visibly* (parity-DVPA), *one-counter* (parity-D1CA), or *visibly one-counter* (parity-DV1CA) respectively, if the underlying DPDM is visibly, one-counter, or both.

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 bounded delay function can be decided. 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 or linear delay is undecidable. 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 a function from $\mathcal{F}$.

As we consider winning conditions L that are recognizable by a 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, 9], we conclude that delay games are also determined.

Remark 1. Let L be recognizable by a parity-DPDA and $f \colon \mathbb{N} \to \mathbb{N}_+$. Then, $\Gamma_f(L)$ is determined.

By applying the result of [10] and 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 1. *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 $\Gamma_f(L(\mathcal{A}))$?*

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 $(0\colon I_0), \dots, (k-2\colon I_{k-2}), (k-1\colon \texttt{STOP})$, where the first entry of a line $(\ell\colon I_\ell)$ 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.

Theorem 2. *The following problem is undecidable:*
Input: *Parity-DPDA $\mathcal{A}$.*
Question: *Does Player O win the game induced by $L(\mathcal{A})$ with finite delay?*

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

Now, define $\mathrm{Conf} = [k](r_0 + r_1)^*$, $\mathrm{Conf}_0 = 0$, and consider the following game specification over the alphabets $\Sigma_I = \{\sharp, r_0, r_1\} \cup [k]$ 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.

Figure 1 depicts the encoding of three configurations (here, $*$ denotes an arbitrary letter). Assuming that line 3 contains $\texttt{INC(X}_0\texttt{)}$ and line 4 contains $\texttt{DEC(X}_1\texttt{)}$, 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.

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

Fig. 1. Part of a play encoding three configurations.

This winning condition can be recognized by a parity-DPDA $\mathcal{A}_\mathcal{R}$: the automaton checks whether the 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 $\ell w \sharp \ell' w'$:

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

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 an error is claimed correctly.

We show that $\mathcal{R}$ halts if and only if Player O wins the game induced by $L(\mathcal{A}_\mathcal{R})$ with finite delay.

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 $\Gamma_f(L(\mathcal{A}_\mathcal{R}))$ that 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 $|y| = 5(i+1)$. Let c and c' denote the last encoding of a configuration in x and the first encoding of a configuration in y, respectively. As Player O has not signaled an error at the previous $\sharp$, we know that c' is well-defined and that it is an 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 the 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 encodings of the infinite run of $\mathcal{R}$ starting in the initial configuration. □

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

Corollary 1. *The following problems are undecidable:*

– ***Input:*** *Parity-DPDA $\mathcal{A}$.*
  Question: *Does Player O win the game induced by $L(\mathcal{A})$ with constant delay?*
– ***Input:*** *Parity-DPDA $\mathcal{A}$ and $k \in \mathbb{N}$.*
  Question: *Does Player O win the game induced by $L(\mathcal{A})$ with linear delay k?*
– ***Input:*** *Parity-DPDA $\mathcal{A}$.*
  Question: *Does Player O win the game induced by $L(\mathcal{A})$ with linear delay?*

By slightly modifying the game described above, one shows that all undecidability results hold even for winning conditions presented by parity-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 $\Sigma_{int} = \Sigma_I \times \{Int, N, L, E_0, E_1\}$, 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. Furthermore, all necessary tests can be implemented using a single stack symbol.

To conclude this section, we give a general criterion to determine whether a set $\mathcal{F}$ of delay functions allows to decide whether Player O wins a given delay game with a 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 $\mathcal{F}$.

Theorem 3. *The following problem is decidable if and only if $\mathcal{F}$ is a bounded set of delay functions:*
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 $f \in \mathcal{F}_{max}$ such that Player O wins $\Gamma_f(L(\mathcal{A}))$. As $\mathcal{F}_{max}$ is finite and every $f \in \mathcal{F}_{max}$ satisfies "$\{i \mid f(i) \neq 1\}$ is finite", the latter property can be decided by Theorem 1.

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 $\Gamma_f(L(\mathcal{A}))$. It follows that there exists 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 $\Gamma_f(L(\mathcal{A}))$ is also 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 2 by allowing Player O to postpone the beginning of the simulation of a computation of $\mathcal{R}$ until she has attained enough delay 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 lines, define Conf_0 and Conf as in the proof of Theorem 2, and consider the following game specification over the alphabets $\Sigma_I = \{\sharp, r_0, r_1, \$\} \cup [k]$ 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 has played 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 2. 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 $d = \sum_{j=1}^{n+1} j$ 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 the 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, start 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. □

Again, the winning condition described in the proof above can be recognized by a parity-DV1CA. Hence, Theorem 3 holds even for such automata.

4 Lower Bounds on Delays

In this section we show that there exists a context-free winning condition L such that Player O wins the game induced by L with finite, but non-elementary delay. To this end, we adapt the idea of the previous section: Player I produces blocks on which a successor relation is defined. 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 4. *There exists a parity-DPDA $\mathcal{A}$ such that Player O wins the game induced by $L(\mathcal{A})$ with finite delay, but for any elementary delay function f the game $\Gamma_f(L(\mathcal{A}))$ is won by Player I.*

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 range $0 \leq j \leq k-1$ if $n_{j+1} \neq 2n_j$ (note that $n_0 = 1$ for every $w_i \in B$). We say that 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 2 depicts the encoding of three blocks in the first component. The blocks w_1 and w_2 constitute a copy error, as w_2 only contains three $\flat$-blocks, and not the required four (due to $|w_1| = 4$). This error was signaled by the letter $\sharp_C$ in front of w_1. Furthermore, the block w_1 contains a doubling error.

$$\begin{matrix}\sharp_N\\ N\end{matrix}\;\overbrace{\begin{matrix}\flat_N & 0\\ E_D & N\end{matrix}}^{w_0}\;\begin{matrix}\sharp_C\\ N\end{matrix}\;\overbrace{\begin{matrix}\flat_N & 0 & \flat_N & 0\\ H & N & N & N\end{matrix}}^{w_1}\;\begin{matrix}\sharp_N\\ N\end{matrix}\;\overbrace{\begin{matrix}\flat_N & 0 & \flat_N & 0 & 0 & \flat_N & 0 & 0 & 0 & 0\\ N & N & N & N & N & N & N & N & N & N\end{matrix}}^{w_2}\;\begin{matrix}\sharp_N\\ N\end{matrix}\;\cdots$$

Fig. 2. A play prefix with three blocks w_0, w_1, and w_2.

Consider the following game specification over the alphabets $\Sigma_I = \{0\} \cup S_\sharp \cup S_\flat$ 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} contains 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 answering every $\flat_{j'}$ for $j' < j$ not by H).

Going back to the example in Figure 2, we see that Player O has claimed the doubling error in blocks w_1 by choosing E_D as second letter and has marked its position by playing H in front if it.

A play of this game is winning for Player O if and only if the induced word $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ satisfies

- $\alpha \notin S_\sharp B_0 (S_\sharp B)^\omega$ (i.e., Player I does not adhere to the format), or
- 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 was answered by H (and ℓ is minimal with this property) and w_i contains a doubling error at position ℓ (i.e., Player O detected a doubling error in block w_i and Player I has not signaled an error in front of a block w_j for $j < i$. Note that the doubling error in block w_i may have been signaled by Player I), or
- 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 detected a copy error in blocks w_i and w_{i+1} and Player I has not signaled an error in front of a block w_j for $j < i$. Note that the copy error in the blocks w_i w_{i+1} may have been signaled by Player I in front of w_i), or
- 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 has signaled a doubling error but not indicated its position correctly), or
- 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 has signaled a copy error without producing one), or
- $\sharp_i = \sharp_N$ for all i (i.e., Player I has never signaled an error).

Hence, the play in Figure 2 is winning for Player O, as her correct claim precedes the signal 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 $\mathcal{A}$: The automaton proceeds in four phases on an ω-word $\rho = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ where $\alpha = \sharp_0 w_0 \sharp_1 w_1 \sharp_2 w_2 \sharp_3 \cdots \in S_\sharp B_0 (S_\sharp B)^\omega$.

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 $\beta(i)$ was found.

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 placing a $\flat_H$ at the appropriate position. If the first $\flat_H$ does not indicate an error correctly (or if w_i does not contain a $\flat_H$), then the doubling error was not signaled and $\mathcal{A}$ accepts ρ. The automaton accepts in phase four if and only if the error indicated by $\sharp_j$ was not found.

Furthermore, $\mathcal{A}$ checks whether the first component is a word in $S_\sharp B_0(S_\sharp B)^\omega$ and whether it contains at least one letter $\sharp_C$ or $\sharp_D$. If it does not, then ρ is accepted. All the tests described in phases three and four can be implemented in terms of a parity-DPDA.

We continue by showing that Player O wins the game induced by L with finite delay. 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 $\sum_{j=0}^{i} f_e(j) - i$ of Player O. At such a position, Player O

has to make a claim about a block which Player I has 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. Vice versa, if Player O claims the doubling error, then he does not introduce a doubling error while completing the block. Then he continues to stick to the input format. In both cases Player O loses, as her claims precede the claims of Player I. □

Using ideas as presented in Section 3 one can show that the language L can even be recognized by a parity-D1CA or by a parity-DVPA.

5 Conclusion

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

Our undecidability results hold even for visibly winning conditions where Player O controls the behavior of the stack. An interesting open question is whether the problem becomes decidable if Player I controls the behavior of the stack. Also, linear delay is necessary in this case, but it is not clear whether it is sufficient.

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, L., 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. Mostowski, A.W.: Games with forbidden positions. Technical report 78, University of Gdańsk, Poland (1991)
10. Walukiewicz, I.: Pushdown processes: games and model-checking. Inf. Comput., 164(2), 234–263 (2001)

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from `http://aib.informatik.rwth-aachen.de/`. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2005-01 * Fachgruppe Informatik: Jahresbericht 2004
2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15 Uwe Naumann: The Complexity of Derivative Computation
2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"
2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete

2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-07 Alexander Nyßen, Horst Lichter: The MeDUSA Reference Manual, Second Edition

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

2008-11 George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows

2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages

2008-13 Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs

2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers

2008-15 Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems

2008-19 Dirk Wilking: Empirical Studies for the Application of Agile Methods to Embedded Systems

2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications

2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices

2009-04 Daniel Klünder: Entwurf eingebetteter Software mit abstrakten Zustandsmaschinen und Business Object Notation

2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs

2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete

2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I

2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs

2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem

2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm

2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs

2009-12 Martin Neuhäußer, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes

2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games

2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV'09)

2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäußer: Compositional Abstraction for Stochastic Systems

2009-16 George B. Mertzios, Derek G. Corneil: Vertex Splitting and the Recognition of Trapezoid Graphs

2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata

2009-18 Paul Hänsch, Michaela Slaats, Wolfgang Thomas: Parametrized Regular Infinite Games and Higher-Order Pushdown Strategies

2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time

2010-03 Holger Krahn: MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering

2010-04 René Wörzberger: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme

2010-05 Daniel Retkowitz: Softwareunterstützung für adaptive eHome-Systeme

2010-06 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Computing maximum reachability probabilities in Markovian timed automata

2010-07 George B. Mertzios: A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms

2010-08 Carsten Otto, Marc Brockschmidt, Christian von Essen, Jürgen Giesl: Automated Termination Analysis of Java Bytecode by Term Rewriting

2010-09 George B. Mertzios, Shmuel Zaks: The Structure of the Intersection of Tolerance and Cocomparability Graphs

2010-10 Peter Schneider-Kamp, Jürgen Giesl, Thomas Ströder, Alexander Serebrenik, René Thiemann: Automated Termination Analysis for Logic Programs with Cut

2010-11 Martin Zimmermann: Parametric LTL Games

2010-12 Thomas Ströder, Peter Schneider-Kamp, Jürgen Giesl: Dependency Triples for Improving Termination Analysis of Logic Programs with Cut

2010-13 Ashraf Armoush: Design Patterns for Safety-Critical Embedded Systems

2010-14	Michael Codish, Carsten Fuhs, Jürgen Giesl, Peter Schneider-Kamp: Lazy Abstraction for Size-Change Termination
2010-15	Marc Brockschmidt, Carsten Otto, Christian von Essen, Jürgen Giesl: Termination Graphs for Java Bytecode
2010-16	Christian Berger: Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles
2010-17	Hans Grönniger: Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten
2010-18	Ibrahim Armaç: Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.