How Much Lookahead is Needed to Win Infinite Games?*

Felix Klein and Martin Zimmermann

Reactive Systems Group, Saarland University, Germany
{klein, zimmermann}@react.uni-saarland.de

Abstract. Delay games are two-player games of infinite duration in which one player may delay her moves to obtain a lookahead on her opponent's moves. For ω-regular winning conditions it is known that such games can be solved in doubly-exponential time and that doubly-exponential lookahead is sufficient.
We improve upon both results by giving an exponential time algorithm and an exponential upper bound on the necessary lookahead. This is complemented by showing ExpTime-hardness of the solution problem and tight exponential lower bounds on the lookahead. Both lower bounds already hold for safety conditions. Furthermore, solving delay games with reachability conditions is shown to be PSpace-complete.

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 a 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. The seminal theorem of Büchi and Landweber [1] states that the winner of an infinite game on a finite arena with an ω-regular winning condition can be determined and a corresponding finite-state winning strategy can be constructed effectively.

Delay Games. In this work, we consider an extension of the classical framework: in a delay game, one player can postpone her moves for some time to obtain a lookahead on her opponent's moves. This allows her to win some games which she would loose without lookahead, e.g., if her first move depends on the third move of her opponent. Nevertheless, there are winning conditions that cannot be won with any finite lookahead, e.g., if her first move depends on every move of her opponent. Delay arises naturally if transmission of data in networks or components equipped with buffers are modeled.

From a more theoretical point of view, uniformization of relations by continuous functions [14, 15] can be expressed and analyzed using delay games. We

* Partially supported by the DFG projects "TriCS" (ZI 1516/1-1) and "AVACS" (SFB/TR 14). The first author was supported by an IMPRS-CS PhD Scholarship.

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 $\tau\colon \Sigma_I^\omega \to \Sigma_O^\omega$. It is winning for her if $(\alpha, \tau(\alpha))$ is contained in the winning condition $L \subseteq \Sigma_I^\omega \times \Sigma_O^\omega$ for every α. If this is the case, we say that τ uniformizes L. In the classical setting, in which the players pick letters in alternation, the n-th letter of $\tau(\alpha)$ depends only on the first n letters of α. A strategy with bounded lookahead, i.e., only finitely many moves are postponed, induces a Lipschitz-continuous function τ (in the Cantor topology on Σ^ω) and a strategy with unbounded lookahead induces a continuous function (equivalently, a uniformly continuous function, as Σ^ω is compact).

Related Work. Hosch and Landweber proved that it is decidable whether a delay game with an ω-regular winning condition can be won with bounded lookahead [8]. Later, Holtmann, Kaiser, and Thomas revisited the problem and showed that if the delaying player wins such a game with unbounded lookahead, then she already wins it with doubly-exponential bounded lookahead, and gave a streamlined decidability proof yielding an algorithm with doubly-exponential running time [7]. Thus, the delaying player does not gain additional power from having unbounded lookahead, bounded lookahead is sufficient.

Going beyond ω-regularity by considering context-free conditions leads to undecidability and non-elementary lower bounds on the necessary lookahead, even for very weak fragments [5]. Nevertheless, there is another extension of the ω-regular conditions where one can prove the analogue of the Hosch-Landweber Theorem: it is decidable whether the delaying player wins a delay game with bounded lookahead, if the winning condition is definable in weak mondadic second order logic with the unbounding quantifier (WMSO+U) [16]. Furthermore, doubly-exponential lookahead is sufficient for such conditions, provided the delaying player wins with bounded lookahead at all. However, bounded lookahead is not always sufficient to win such games, i.e., the analogue of the Holtmann-Kaiser-Thomas Theorem does not hold for WMSO+U conditions. Finally, all delay games with Borel winning conditions are determined [11].

Stated in terms of uniformization, Hosch and Landweber proved decidability of the uniformization problem for ω-regular relations by Lipschitz-continuous functions and Holtmann et al. proved the equivalence of the existence of a continuous uniformization function and the existence of a Lipschitz-continuous uniformization function for ω-regular relations. Furthermore, uniformization of context-free relations is undecidable, even with respect to Lipschitz-continuous functions, but uniformization of WMSO+U relations by Lipschitz-continuous functions is decidable.

Furthermore, Carayol and Löding considered the case of finite words [3], and Löding and Winter [12] considered the case of finite trees, which are both decidable. However, the non-existence of MSO-definable choice functions on the infinite binary tree [2, 6] implies that uniformization fails for such trees.

Although several extensions of ω-regular winning conditions for delay games have been considered, many problems remain open even for ω-regular conditions: there are no non-trivial lower bounds on the necessary lookahead and on

the complexity of solving such games. Furthermore, only deterministic parity automata were used to specify winning conditions, and the necessary lookahead and the solution complexity is measured in their size. Thus, it is possible that considering weaker automata models like reachability or safety automata leads to smaller lookahead requirements and faster algorithms.

Our Contribution. We answer all these questions and improve upon both results of Holtmann et al. by determining the exact complexity of ω-regular delay games and by giving tight bounds on the necessary lookahead.

First, we present an exponential time algorithm for solving delay games with ω-regular winning conditions, an exponential improvement over the original doubly-exponential time algorithm. Both algorithms share some similarities: given a deterministic parity automaton $\mathcal{A}$ recognizing the winning condition of the game, a parity game is constructed that is won by by the delaying player if and only if she wins the delay game with winning condition $L(\mathcal{A})$. Furthermore, both parity games are induced by equivalence relations that capture the behavior of $\mathcal{A}$. However, our parity game is of exponential size while the one of Holtmann et al. is doubly-exponential. Also, they need an intermediate game, the so-called block game, to prove the equivalence of the delay game and the parity game, while our equivalence proof is direct. Thus, our algorithm and its correctness proof are even simpler than the ones of Holtmann et al.

Second, we show that solving delay games is ExpTime-complete by proving the first non-trivial lower bound on the complexity of ω-regular delay games. The lower bound is proved by a reduction from the acceptance problem for alternating polynomial space Turing machines [4], which results in delay games with safety conditions. Thus, solving delay games with safety conditions is already ExpTime-hard. Our reduction is inspired by the ExpTime-hardness proof for continuous simulation games [9], a simulation game on Büchi automata where Duplicator is able to postpone her moves to obtain a lookahead on Spoiler's moves. However, this reduction is from a two-player tiling problem while we directly reduce from alternating Turing machines.

Third, we determine the exact amount of lookahead necessary to win delay games with ω-regular conditions. From our algorithm we derive an exponential upper bound, which is again an exponential improvement. This upper bound is complemented by the first non-trivial lower bound on the necessary lookahead: there are reachability and safety conditions that are winning for the delaying player, but only with exponential lookahead, i.e., our upper bound is tight.

Fourth, we present the first results for fragments of ω-regular conditions. As already mentioned above, our lower bounds on complexity and necessary lookahead already hold for safety conditions, i.e., safety is already as hard as parity. Thus, the complexity of the problems manifests itself in the transition structure of the automaton, not in the acceptance condition. For reachability conditions, the situation is different: we show that solving delay games with reachability conditions is equivalent to universality of non-deterministic reachability automata and therefore PSpace-complete.

Omitted proofs can be found in the full version [10].

2 Preliminaries

The non-negative integers are denoted by $\mathbb{N}$. An alphabet Σ is a non-empty finite set, Σ^* the set of finite words over Σ, Σ^n the set of words of length n, and Σ^ω the set of infinite words. The empty word is denoted by ε and the length of a finite word w by $|w|$. For $w \in \Sigma^* \cup \Sigma^\omega$ we write $w(n)$ for the n-th letter of w.

Automata. We use automata of the form $\mathcal{A} = (Q, \Sigma, q_I, \Delta, \varphi)$ where $\Delta\colon Q \times \Sigma \to 2^Q \setminus \{\emptyset\}$ is a a non-deterministic transition function and where the acceptance condition φ is either a set $F \subseteq Q$ of accepting states or a coloring $\Omega\colon Q \to \mathbb{N}$. An automaton is deterministic, if $|\Delta(q,a)| = 1$ for every q and a. In this case, we denote Δ by a function $\delta\colon Q \times \Sigma \to Q$. A state q of $\mathcal{A}$ is a sink, if $\Delta(q,a) = \{q\}$ for every $a \in \Sigma$. Finite and infinite runs are defined as usual. Given an automaton $\mathcal{A}$ over Σ with some set F of accepting states or with some coloring Ω, we consider the following acceptance modes:

Finite: $L_*(\mathcal{A}) \subseteq \Sigma^*$ denotes the set of finite words accepted by $\mathcal{A}$, i.e., the set of words that have a run ending in F.

Reachability: $L_\exists(\mathcal{A}) \subseteq \Sigma^\omega$ denotes the set of infinite words that have a run visiting an accepting state at least once. We have $L_\exists(\mathcal{A}) = L_*(\mathcal{A}) \cdot \Sigma^\omega$.

Safety: Dually, $L_\forall(\mathcal{A}) \subseteq \Sigma^\omega$ denotes the set of infinite words that have a run only visiting accepting states.

Parity: $L_p(\mathcal{A}) \subseteq \Sigma^\omega$ denotes the set of infinite words that have a run such that the maximal color visited infinitely often during this run is even.

Note that we require automata to be complete. For safety and parity acceptance this is no restriction, since we can always add a fresh rejecting sink and lead all missing transitions to this sink. However, incomplete automata with reachability acceptance are strictly stronger than complete ones, as incompleteness can be used to check safety properties. We impose this restriction since we are interested in pure reachability conditions.

Given a language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$ we denote by $\mathrm{pr}_I(L)$ its projection to the first component. Similarly, given an automaton $\mathcal{A}$ over $\Sigma_I \times \Sigma_O$, we denote by $\mathrm{pr}_I(\mathcal{A})$ the automaton obtained by projecting each letter to its first component.

Remark 1. Let $\mathrm{acc} \in \{*, \exists, \forall, p\}$, then $\mathrm{pr}_I(L_{\mathrm{acc}}(\mathcal{A})) = L_{\mathrm{acc}}(\mathrm{pr}_I(\mathcal{A}))$.

Games with Delay. A delay function is a mapping $f\colon \mathbb{N} \to \mathbb{N} \setminus \{0\}$, which is said to be constant, if $f(i) = 1$ for every $i > 0$. Given an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$ and a delay function f, the game $\Gamma_f(L)$ is played by two players, the input player "Player I" and the output player "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 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 the outcome $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ is in L, otherwise Player I wins.

Given a delay function f, a strategy for Player I is a mapping $\tau_I\colon \Sigma_O^* \to \Sigma_I^*$ where $|\tau_I(w)| = f(|w|)$, 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 $i \in \mathbb{N}$. It is consistent with τ_O, if $v_i = \tau_O(u_0 \cdots u_i)$ for every $i \in \mathbb{N}$. A strategy τ for Player $p \in \{I, O\}$ is winning, if every play that is consistent with τ is winning for Player p. We say that a player wins $\Gamma_f(L)$, if she has a winning strategy.

Example 1. Consider L over $\{a, b, c\} \times \{b, c\}$ with $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)} \cdots \in L$, if $\alpha(n) = a$ for every $n \in \mathbb{N}$ or if $\beta(0) = \alpha(n)$, where n is the smallest position with $\alpha(n) \neq a$. Intuitively, Player O wins, if the letter she picks in the first round is equal to the first letter other than a that Player I picks. Also, Player O wins, if there is no such letter. Note that L can be accepted by a safety automaton.

We claim that Player I wins $\Gamma_f(L)$ for every delay function f: Player I picks $a^{f(0)}$ in the first round and assume Player O picks b afterwards (the case where she picks c is dual). Then, Player I picks a word starting with c in the second round. The resulting play is winning for Player I no matter how it is continued. Thus, Player I has a winning strategy in $\Gamma_f(L)$.

Note that if a language L is recognizable by a (deterministic) parity automaton, then $\Gamma_f(L)$ is determined, as a delay game with parity condition can be expressed as an explicit parity game in a countable arena.

Also, note that universality of $\mathrm{pr}_I(L)$ is a necessary condition for Player O to win $\Gamma_f(L)$. Otherwise, Player I could pick a word from $\Sigma_I^\omega \setminus \mathrm{pr}_I(L)$, which is winning for him, no matter how Player O responds.

Proposition 1. *If Player O wins $\Gamma_f(L)$, then $\mathrm{pr}_I(L)$ is universal.*

3 Lower Bounds on the Lookahead

In this section, we prove lower bounds on the necessary lookahead for Player O to win delay games with reachability or safety conditions. Thus, the same bounds hold for more expressive conditions like Büchi, co-Büchi, and parity. They are complemented by an exponential upper bound for parity conditions in the next section. Note that both lower bounds already hold for deterministic automata.

Theorem 1. *For every $n > 1$ there is a language L_n such that*

- *$L_n = L_\exists(\mathcal{A}_n)$ for some deterministic automaton $\mathcal{A}_n$ with $|\mathcal{A}_n| \in \mathcal{O}(n)$,*
- *Player O wins $\Gamma_f(L_n)$ for some constant delay function f, but*
- *Player I wins $\Gamma_f(L_n)$ for every delay function f with $f(0) \leq 2^n$.*

Proof. Let $\Sigma_I = \Sigma_O = \{1, \ldots, n\}$. We say that w in Σ_I^* contains a bad j-pair, for $j \in \Sigma_I$, if there are two occurrences of j in w such that no $j' > j$ occurs in between. The automaton $\mathcal{B}_j$, depicted in Figure 1(a), accepts exactly the words with a bad j-pair. Now, consider the language L over Σ_I defined by

$$L = \bigcap_{1 \leq j \leq n} \{w \in \Sigma_I^* \mid w \text{ contains no bad } j\text{-pair}\}.$$

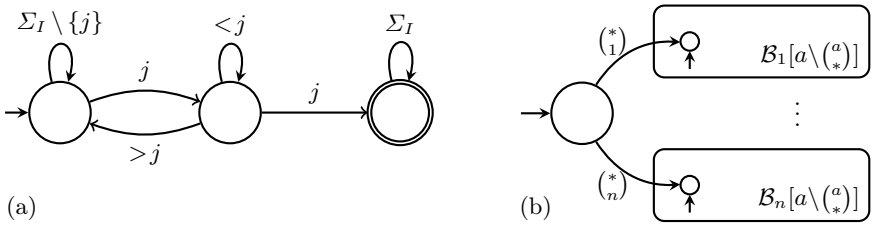

Fig. 1. (a) Automaton $\mathcal{B}_j$ for $j \in \Sigma_I$. (b) Construction of $\mathcal{A}_n$.

Straightforward inductions show that every $w \in L$ satisfies $|w| < 2^n$ and that there is a word $w_n \in L$ with $|w_n| = 2^n - 1$.

The winning condition L_n is defined as follows: $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots$ is in L_n if $\alpha(1)\alpha(2)\cdots$ contains a bad $\beta(0)$-pair, i.e., with her first move, Player O has to pick a j such that Player I has produced a bad j-pair. For technical reasons, the first letter picked by Player I is ignored. The construction of an automaton $\mathcal{A}_n$ recognizing L_n is sketched in Figure 1(b). Here, $\mathcal{B}_j[a\backslash\binom{a}{*}]$ denotes $\mathcal{B}_j$, where for each $a \in \Sigma_I$ every transition labeled by a is replaced by transitions labeled by $\binom{a}{b}$ for every $b \in \Sigma_O$. Clearly, we have $\mathcal{A}_n \in \mathcal{O}(n)$.

Player O wins $\Gamma_f(L_n)$ for every delay function with $f(0) > 2^n$. In the first round, Player I has to pick a word u_0 such that u_0 without its first letter is not in L. This allows Player O to find a bad j-pair for some j, i.e., she wins the play no matter how it is continued.

However, for f with $f(0) \leq 2^n$, Player I has a winning strategy by picking the prefix of $1w_n$ of length $f(0)$ in the first round. Player O has to answer with some $j \in \Sigma_O$. In this situation, Player I can continue by finishing w_n and then playing some $j' \neq j$ ad infinitum, which ensures that the resulting sequence does not contain a bad j-pair. Thus, the play is winning for Player I. □

Using a similar construction, one can show exponential lower bounds for safety conditions as well.

Theorem 2. *For every $n > 1$ there is a language L'_n such that*

- *$L'_n = L_\forall(\mathcal{A}'_n)$ for some deterministic automaton $\mathcal{A}'_n$ with $|\mathcal{A}'_n| \in \mathcal{O}(n)$,*
- *Player O wins $\Gamma_f(L'_n)$ for some constant delay function f, but*
- *Player I wins $\Gamma_f(L'_n)$ for every delay function f with $f(0) \leq 2^n$.*

The aforementioned constructions also work for constant-size alphabets, if we encode every $j \in \{1, \ldots, n\}$ in binary, resulting in automata $\mathcal{A}_n$ and $\mathcal{A}'_n$ whose sizes are in $\mathcal{O}(n \log n)$. It is open whether linear-sized automata and a constant-sized alphabet can be achieved simultaneously.

4 Computational Complexity of Delay Games

In this section, we determine the computational complexity of solving delay games. First, we consider the special case of reachability conditions and prove

such games to be PSPACE-complete. Then, we show that games with safety conditions are EXPTIME-hard. The latter bound is complemented by an EXPTIME-algorithm for solving delay games with parity conditions. From this algorithm, we also deduce an exponential upper bound on the necessary lookahead for Player O, which matches the lower bounds given in the previous section.

4.1 Reachability Conditions

Recall that universality of the projection to the first component of the winning condition is a necessary condition for Player O for having a winning strategy in a delay game. Our first result in this section states that universality is also sufficient in the case of reachability conditions. Thus, solving delay games with reachability conditions is equivalent, via linear time reductions, to the universality problem for non-deterministic reachability automata, which is PSPACE-complete. Also, our proof yields an exponential upper bound on the necessary lookahead.

Theorem 3. *Let $L = L_\exists(\mathcal{A})$, where $\mathcal{A}$ is a non-deterministic reachability automaton. The following are equivalent:*

1. *Player O wins $\Gamma_f(L)$ for some delay function f.*
2. *Player O wins $\Gamma_f(L)$ for some constant delay function f with $f(0) \leq 2^{|\mathcal{A}|}$.*
3. *$\mathrm{pr}_I(L)$ is universal.*

Proof. To show the equivalence, it only remains to prove 3. $\Rightarrow$ 2.

We assume w.l.o.g. that the accepting states of $\mathcal{A}$ are sinks, which implies that $L_*(\mathrm{pr}_I(\mathcal{A}))$ is suffix-closed, i.e., $w \in L_*(\mathrm{pr}_I(\mathcal{A}))$ implies $ww' \in L_*(\mathrm{pr}_I(\mathcal{A}))$ for every $w' \in \Sigma_I^*$. Furthermore, let $\mathcal{A}^c$ be an automaton recognizing the complement of $L_*(\mathrm{pr}_I(\mathcal{A}))$, which is prefix-closed, as it is the complement of a suffix-closed language. We can choose $\mathcal{A}^c$ such that $|\mathcal{A}^c| \leq 2^{|\mathcal{A}|}$.

We claim that $L_*(\mathcal{A}^c)$ is finite. Assume it is infinite. Then, by König's Lemma there is an infinite word α whose prefixes are all in $L_*(\mathcal{A}^c)$. Due to universality, we have $\alpha \in L_\exists(\mathrm{pr}_I(\mathcal{A}))$, i.e., there is a prefix of α in $L_*(\mathrm{pr}_I(\mathcal{A}))$. Thus, the prefix is in $L_*(\mathrm{pr}_I(\mathcal{A}))$ and in the complement $L_*(\mathcal{A}^c)$ yielding the desired contradiction. An automaton with n states with a finite language accepts words of length at most $n-1$. Thus, $w \in L_*(\mathrm{pr}_I(\mathcal{A}))$ for every $w \in \Sigma_I^*$ with $|w| \geq 2^{|\mathcal{A}|}$.

Using this, we show that Player O wins $\Gamma_f(L)$ if $f(0) = 2^{|\mathcal{A}|}$. Player I has to pick $f(0)$ letters with his first move, say $u_0 = \alpha(0) \cdots \alpha(f(0)-1)$. As $f(0)$ is large enough, we have $u_0 \in L_*(\mathrm{pr}_I(\mathcal{A}))$. Hence, there is a word $\beta(0) \cdots \beta(f(0)-1) \in \Sigma_O^*$ such that $\binom{\alpha(0)}{\beta(0)} \cdots \binom{\alpha(f(0)-1)}{\beta(f(0)-1)} \in L_*(\mathcal{A})$. By picking $\beta(0), \ldots, \beta(f(0)-1)$ in the first $f(0)$ rounds, Player O wins the play, no matter how it is continued. Hence, she has a winning strategy. □

As universality for non-deterministic reachability automata is PSPACE-complete (see, e.g., [10]), we obtain the following consequence of Theorem 3.

Corollary 1. *The following problem is* PSPACE*-complete: Given a non-deterministic reachability automaton $\mathcal{A}$, does Player O win $\Gamma_f(L_\exists(\mathcal{A}))$ for some f?*

The upper bounds on complexity and necessary lookahead hold for non-deterministic automata while the lower bounds hold for deterministic ones [10].

4.2 Safety Conditions

Unsurprisingly, Example 1 shows that Theorem 3 does not hold for safety conditions: the projection $\mathrm{pr}_I(L)$ is universal, but Player O has no winning strategy for any delay function. It turns out that safety conditions are even harder than reachability conditions (unless PSpace equals ExpTime).

Theorem 4. *The following problem is* ExpTime*-hard: Given a deterministic safety automaton $\mathcal{A}$, does Player O win $\Gamma_f(L_\forall(\mathcal{A}))$ for some f?*

The proof proceeds by a reduction from the non-acceptance problem for alternating polynomial space Turing machines, which is sufficient due to APSpace = ExpTime [4] being closed under complement. Fix such a machine $\mathcal{M}$, an input x, and a polynomial p that bounds the space consumption of $\mathcal{M}$. We construct a safety automaton $\mathcal{A}$ of polynomial size in $|\mathcal{M}| + p(|x|)$ such that $\mathcal{M}$ rejects x if and only if Player O wins $\Gamma_f(L_\forall(\mathcal{A}))$ for some f. To this end, we give Player I control over the existential states while Player O controls the universal ones. Additionally, Player I is in charge of producing all configurations with his moves. He can copy configurations in order to wait for Player O's decisions of successors for universal transitions, which are delayed due to the lookahead.

More formally, the input alphabet Σ_I consists of the alphabet and the set of states of $\mathcal{M}$ and of two separators N and C while the output alphabet Σ_O consists of the transition relation of $\mathcal{M}$ and of two signals ✗ and ✓. Intuitively, Player I produces configurations of $\mathcal{M}$ of length $p(|x|)$ preceded by either C or N to denote whether the configuration is a *copy* of the previous one or a *new* one. Copying configurations is necessary to bridge the lookahead while waiting for Player O to determine the transition that is applied to a universal configuration. Player I could copy a configuration ad infinitum, but this will be losing for him, unless it is an accepting one. Player O chooses universal transitions at every separator[1] N by picking a transition of $\mathcal{M}$. At every other position, she has to pick a signal: ✗ allows her to claim an error in the configurations picked by Player O while ✓ means that she does not claim an error at the current position.

The automaton $\mathcal{A}$ checks that Player I produces only legal configurations, that he starts with the initial one, that Player O always picks a transition at the separators, and that the first error claimed by Player O is indeed an error. If it is one, then $\mathcal{A}$ goes to an accepting sink, otherwise to a rejecting sink. Finally, if Player I produces an accepting configuration without Player O correctly claiming an error in a preceding configuration, then $\mathcal{A}$ goes to a rejecting sink.

These properties can be checked by a deterministic safety automaton of polynomial size, as the configuations of $\mathcal{M}$ are of polynomial size. A detailed description of $\mathcal{A}$ and a proof that $\mathcal{M}$ rejects x if and only if Player O wins $\Gamma_f(L_\forall(\mathcal{A}))$ for some f can be found in the full version [10].

It is noteworthy that the ExpTime lower bound does not require the full exponential lookahead that might be necessary to win delay games with safety

[1] If the following configuration is existential or the separator is a C, then her choice is ignored.

conditions: Player O wins the game constructed above with constant lookahead that is smaller than $|\mathcal{A}|$, if she wins at all.

4.3 Parity Conditions

Now, we complement the ExpTime lower bound shown in the previous subsection with an exponential time algorithm for solving delay games with parity conditions. Thus, delay games with safety or parity conditions are ExpTime-complete. Also, we derive an exponential upper bound on the necessary lookahead from the algorithm. All results only hold for deterministic automata.

Theorem 5. *The following problem is in* ExpTime*: Given a deterministic parity automaton $\mathcal{A}$, does Player O win $\Gamma_f(L_p(\mathcal{A}))$ for some delay function f?*

We begin by constructing an exponentially-sized, delay-free parity game with the same number of colors as $\mathcal{A}$, which is won by Player O if and only if she wins $\Gamma_f(L_p(\mathcal{A}))$ for some delay function f.

Let $\mathcal{A} = (Q, \Sigma_I \times \Sigma_O, q_I, \delta, \Omega)$ with $\Omega\colon Q \to \mathbb{N}$. First, we adapt $\mathcal{A}$ to keep track of the maximal color visited during a run. To this end, we define the automaton $\mathcal{C} = (Q_\mathcal{C}, \Sigma_I \times \Sigma_O, q_I^\mathcal{C}, \delta_\mathcal{C}, \Omega_\mathcal{C})$ where $Q_\mathcal{C} = Q \times \Omega(Q)$, $q_I^\mathcal{C} = (q_I, \Omega(q_I))$,

$$\delta_\mathcal{C}((q,c),a) = (\delta(q,a), \max\{c, \Omega(\delta(q,a))\}),$$

and $\Omega_\mathcal{C}(q,c) = c$. We denote the size of $\mathcal{C}$ by n. Note that $\mathcal{C}$ does not recognize $L_p(\mathcal{A})$. However, we are only interested in runs on finite play infixes.

Remark 2. Let $w \in (\Sigma_I \times \Sigma_O)^*$ and let $(q_0,c_0)(q_1,c_1)\cdots(q_{|w|},c_{|w|})$ be the run of $\mathcal{C}$ on w from some state $(q_0,c_0) \in \{(q,\Omega(q)) \mid q \in Q\}$. Then, $q_0q_1\cdots q_{|w|}$ is the run of $\mathcal{A}$ on w starting in q_0 and $c_{|w|} = \max\{\Omega(q_j) \mid 0 \le j \le |w|\}$.

In the following, we work with partial functions from $Q_\mathcal{C}$ to $2^{Q_\mathcal{C}}$, where we denote the domain of such a function r by $\mathrm{dom}(r)$. Intuitively, we use r to capture the information encoded in the lookahead provided by Player I. Assume Player I has picked $\alpha(0)\cdots\alpha(j)$ and Player O has picked $\beta(0)\cdots\beta(i)$ for $i < j$ such that the lookahead is $w = \alpha(i+1)\cdots\alpha(j)$. Then, we can determine the state q that $\mathcal{C}$ reaches after processing $\binom{\alpha(0)}{\beta(0)}\cdots\binom{\alpha(i)}{\beta(i)}$, but the automaton cannot process w, since Player O has not yet picked $\beta(i+1)\cdots\beta(j)$. However, we can determine the states Player O can enforce by picking an appropriate completion, which will be the ones contained in $r(q)$. Note that the function r depends on the lookahead w picked by Player I.

To formalize the functions capturing the lookahead picked by Player I, we define $\delta_\mathcal{P}\colon 2^{Q_\mathcal{C}} \times \Sigma_I \to 2^{Q_\mathcal{C}}$ via $\delta_\mathcal{P}(S,a) = \bigcup_{q\in S}\bigcup_{b\in\Sigma_O}\delta_\mathcal{C}(q,\binom{a}{b})$, i.e., $\delta_\mathcal{P}$ is the transition function of the powerset automaton of $\mathrm{pr}_I(\mathcal{C})$. As usual, we extend $\delta_\mathcal{P}$ to $\delta_\mathcal{P}^*\colon 2^{Q_\mathcal{C}} \times \Sigma_I^* \to 2^{Q_\mathcal{C}}$ via $\delta_\mathcal{P}^*(S,\varepsilon) = S$ and $\delta_\mathcal{P}^*(S,wa) = \delta_\mathcal{P}(\delta_\mathcal{P}^*(S,w),a)$.

Let $D \subseteq Q_\mathcal{C}$ be non-empty and let $w \in \Sigma_I^*$. We define the function r_w^D with domain D as follows: for every $(q,c) \in D$, we have

$$r_w^D(q,c) = \delta_\mathcal{P}^*(\{(q,\Omega(q))\}, w).$$

Note that we apply $\delta_{\mathcal{P}}$ to $\{(q, \Omega(q))\}$, i.e., the second argument is the color of q and not the color c from the argument to r_w^D. If $(q', c') \in r_w^D(q, c)$, then there is a word w' whose projection is w and such that the run of $\mathcal{A}$ on w' leads from q to q' and has maximal color c'. Thus, if Player I has picked the lookahead w, then Player O could pick an answer such that the combined word leads $\mathcal{A}$ from q to q' with minimal color c'.

We call w a witness for a partial function $r\colon Q_{\mathcal{C}} \to 2^{Q_{\mathcal{C}}}$, if we have $r = r_w^{\mathrm{dom}(r)}$. Thus, we obtain a language $W_r \subseteq \Sigma_I^*$ of witnesses for each such function r. We define $\mathfrak{R} = \{r \mid \mathrm{dom}(r) \neq \emptyset \text{ and } W_r \text{ is infinite}\}$.

Remark 3. Let $\mathfrak{R}$ be defined as above.

1. Let $r \in \mathfrak{R}$. Then, $r(q) \neq \emptyset$ for every $q \in \mathrm{dom}(r)$.
2. Let r be a partial function from $Q_{\mathcal{C}}$ to $2^{Q_{\mathcal{C}}}$. Then, W_r is recognized by a deterministic finite automaton with 2^{n^2} states.
3. Let $D \subseteq Q_{\mathcal{C}}$ be non-empty and let w be such that $|w| \geq 2^{n^2}$. Then, there exists some $r \in \mathfrak{R}$ with $\mathrm{dom}(r) = D$ and $w \in W_r$.

Now, we can define an abstract game $\mathcal{G}(\mathcal{A})$ which is played between Player I and Player O in rounds $i = 0, 1, 2, \ldots$ as follows: in each round, Player I picks a function from $\mathfrak{R}$ and Player O answers by a state of $\mathcal{C}$ subject to the following constraints. In the first round, Player I has to pick $r_0 \in \mathfrak{R}$ such that $\mathrm{dom}(r_0) = \{q_I^{\mathcal{C}}\}$ (C1) and Player O has to answer by picking a state $q_0 \in \mathrm{dom}(r_0)$, which implies $q_0 = q_I^{\mathcal{C}}$. Now, consider round $i > 0$: Player I has picked functions $r_0, r_1, \ldots, r_{i-1}$ and Player O has picked states $q_0, q_1, \ldots, q_{i-1}$. Now, Player I has to pick a function $r_i \in \mathfrak{R}$ such that $\mathrm{dom}(r_i) = r_{i-1}(q_{i-1})$ (C2). Then, Player O picks some state $q_i \in \mathrm{dom}(r_i)$.

Both players can always move: Player I can move, as $r_{i-1}(q_{i-1})$ is always non-empty (Remark 3.1) and thus the domain of some $r \in \mathfrak{R}$ (Remark 3.3), and Player O can move, as the domain of every $r \in \mathfrak{R}$ is non-empty by construction. The resulting play of $\mathcal{G}(\mathcal{A})$ is the sequence $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$. It is won by Player O if the maximal color occurring infinitely often in $\Omega_{\mathcal{C}}(q_0)\Omega_{\mathcal{C}}(q_1)\Omega_{\mathcal{C}}(q_2)\cdots$ is even. Otherwise, Player I wins.

A strategy for Player I is a function τ_I' mapping the empty play prefix to a function r_0 satisfying (C1) and mapping a non-empty prefix $r_0 q_0 \cdots r_{i-1} q_{i-1}$ to a function r_i satisfying (C2). A strategy for Player O maps a play prefix $r_0 q_0 \cdots r_i$ to a state $q_i \in \mathrm{dom}(r_i)$. A play $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$ is consistent with τ_I', if $r_i = \tau_I'(r_0 q_0 \cdots r_{i-1} q_{i-1})$ for every $i \in \mathbb{N}$ and it is consistent with τ_O', if $q_i = \tau_O'(r_0 q_0 \cdots r_i)$ for every $i \in \mathbb{N}$. A strategy τ' for Player $p \in \{I, O\}$ is winning, if every play that is consistent with τ' is winning for Player p. As usual, we say that a player wins $\mathcal{G}(\mathcal{A})$, if she has a winning strategy.

Lemma 1. *Player O wins $\Gamma_f(L_p(\mathcal{A}))$ for some delay function f if and only if Player O wins $\mathcal{G}(\mathcal{A})$.*

Now, we can prove Theorem 5. Due to Lemma 1, we just have to show that we can construct and solve an explicit version of $\mathcal{G}(\mathcal{A})$ in exponential time.

Proof. First, we argue that $\mathfrak{R}$ can be constructed in exponential time: to this end, one constructs for every partial function r from $Q_{\mathcal{C}}$ to $2^{Q_{\mathcal{C}}}$ the automaton of Remark 3.2 recognizing W_r and tests it for recognizing an infinite language. There are exponentially many functions and each automaton is of exponential size, which yields the desired result. To conclude, we encode $\mathcal{G}(\mathcal{A})$ as a graph-based parity game of exponential size with the same number of colors as $\mathcal{A}$. Such a game can be solved in exponential time in the size of $\mathcal{A}$ [13]. □

The proof of Lemma 1 yields the exponential upper bound $2^{(|\mathcal{A}|k)^2+1}$ on the necessary lookahead, where k is the number of colors of $\mathcal{A}$. However, this can be improved by using a direct pumping argument.

Theorem 6. *Let $L = L_p(\mathcal{A})$ where $\mathcal{A}$ is a deterministic parity automaton with k colors. The following are equivalent:*

1. *Player O wins $\Gamma_f(L)$ for some delay function f.*
2. *Player O wins $\Gamma_f(L)$ for some constant delay function f with $f(0) \leq 2^{2|\mathcal{A}|k+2} + 2$.*

5 Conclusion

We gave the first algorithm that solves ω-regular delay games in exponential time, which is an exponential improvement over the previously known algorithms. We complemented this by showing the problem to be ExpTime-complete, even for safety conditions. Also, we determined the exact amount of lookahead that is necessary to win ω-regular delay games by proving tight exponential bounds, which already hold for safety and reachability conditions. Finally, we showed solving games with reachability conditions to be PSpace-complete. To the best of our knowledge, all lower bounds are the first non-trivial ones for delay games.

Our lower bounds already hold for deterministic automata while our upper bounds (but the ones for reachability) only hold for deterministic automata. One can obviously obtain upper bounds for non-deterministic automata via determinization, but this incurs an exponential blowup, which might not be optimal. We leave the study of this problem for future work. Another open question concerns the influence of using different deterministic automata models that recognize the class of ω-regular conditions, e.g., Rabin, Streett, and Muller automata, on the necessary lookahead and the solution complexity, again measured in the size of the automata. Indeed, our construction used to prove Theorem 5 can be adapted to deal with these acceptance conditions, e.g., for conditions given by Muller automata, $\mathcal{C}$ keeps track of the vertices visited on a run and $\mathcal{G}(\mathcal{A})$ is a Muller game. This yields upper bounds, but it is open whether these are optimal.

Finally, we also considered winning conditions that are both reachability and safety conditions. Here, polynomial lookahead suffices and the problem is in Π_2^{P}, i.e., in the second level of the polynomial hierarchy. Again, both results only hold for deterministic automata and are presented in the full version [10]. In future work, we aim to find lower bounds.

Acknowledgments. We thank Bernd Finkbeiner for a fruitful discussion that lead to Theorem 1 and Theorem 3.

References

1. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Trans. Amer. Math. Soc. 138, pp. 295–311 (1969)
2. Carayol, A., Löding, C.: MSO on the infinite binary tree: Choice and order. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 161–176. Springer (2007)
3. Carayol, A., Löding, C.: Uniformization in automata theory. In: Schroeder-Heister, P., Heinzmann, G., Hodges, W., Bour, P.E. (eds.) CLMPS. College Publications, London (2012), to appear
4. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. J. ACM 28(1), 114–133 (1981)
5. Fridman, W., Löding, C., Zimmermann, M.: Degrees of lookahead in context-free infinite games. In: Bezem, M. (ed.) CSL 2011. LIPIcs, vol. 12, pp. 264–276. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2011)
6. Gurevich, Y., Shelah, S.: Rabin's uniformization problem. The Journal of Symbolic Logic 48, 1105–1119 (1983)
7. Holtmann, M., Kaiser, L., Thomas, W.: Degrees of lookahead in regular infinite games. LMCS 8(3) (2012)
8. Hosch, F.A., Landweber, L.H.: Finite delay solutions for sequential conditions. In: ICALP 1972. pp. 45–60 (1972)
9. Hutagalung, M., Lange, M., Lozes, É.: Buffered simulation games for Büchi automata. In: Ésik, Z., Fülöp, Z. (eds.) AFL 2014. EPTCS, vol. 151, pp. 286–300 (2014)
10. Klein, F., Zimmermann, M.: How much lookahead is needed to win infinite games? ArXiv:1412.3701 (2014)
11. Klein, F., Zimmermann, M.: What are strategies in delay games? Borel determinacy for games with lookahead. ArXiv: 1504.02627 (2015)
12. Löding, C., Winter, S.: Synthesis of deterministic top-down tree transducers from automatic tree relations. In: Peron, A., Piazza, C. (eds.) GandALF 2014. EPTCS, vol. 161, pp. 88–101 (2014)
13. Schewe, S.: Solving parity games in big steps. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 449–460. Springer (2007)
14. Thomas, W., Lescow, H.: Logical specifications of infinite computations. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.) A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium. LNCS, vol. 803, pp. 583–621. Springer (1993)
15. Trakhtenbrot, B., Barzdin, I.: Finite Automata; Behavior and Synthesis. Fundamental Studies in Computer Science, V. 1, North-Holland Publishing Company; New York: American Elsevier (1973)
16. Zimmermann, M.: Delay games with WMSO+U winning conditions. In: CSR 2015 (2015), to appear. Available at arxiv.org/abs/1412.3978