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 by 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 [2,3] 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 the second player 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 (or 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 ω-regular winning condition can be won with bounded lookahead [4]. 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 with doubly-exponential bounded lookahead, and gave a streamlined decidability proof yielding an algorithm with doubly-exponential running time [5]. 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 [6]. 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) [7]. Furthermore, doubly-exponential lookahead is sufficient for such conditions, provided the delaying player wins with bounded lookahead at all. This leaves open the possibility that there are WMSO+U conditions that require unbounded lookahead, i.e., it is open whether the analogue of the Holtmann-Kaiser-Thomas Theorem holds for WMSO+U conditions.

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.

In another line of work, Carayol and Löding considered the case of finite words [8], and Löding and Winter [9] considered the case of finite trees, which are both decidable. However, Carayol and Löding showed that uniformization over infinite trees fails [10].

Although several extensions of ω-regular winning conditions for delay games have been considered, many problems remain open even for ω-regular condi-

tions: 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 Player O 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 solving delay games to be ExpTime-complete by proving the first non-trivial lower bound on the complexity of ω-regular delay games. The lower bound is proven by a reduction from the acceptance problem for alternating polynomial space Turing machines [11], 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 [12], 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 reduce directly 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 automata and therefore PSpace-complete.

2 Preliminaries

The set of non-negative integers is denoted by $\mathbb{N}$. An alphabet Σ is a non-empty finite set of letters, 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.

2.1 Automata

An automaton $\mathcal{A} = (Q, \Sigma, q_I, \Delta, \varphi)$ consists of a finite set Q of states with initial state $q_I \in Q$, a alphabet Σ, a non-deterministic transition function $\Delta\colon Q \times \Sigma \to 2^Q \setminus \{\emptyset\}$, and an acceptance condition φ, which is either a set $F \subseteq Q$ of accepting states, depicted by doubly-lined states, or a coloring $\Omega\colon Q \to \mathbb{N}$. Note that we require automata to be complete, i.e., every state has at least one outgoing transition labeled by a for every $a \in \Sigma$. We discuss this restriction after introducing the different acceptance modes. The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the cardinality of Q. A run of $\mathcal{A}$ on a finite or infinite word α over Σ is a path through $\mathcal{A}$, starting in q_I, whose transitions are labeled by α.

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 deterministic automaton has a unique run on every finite or infinite word. A state q of $\mathcal{A}$ is a sink state, if $\Delta(q, a) = \{q\}$ for every $a \in \Sigma$.

Given an automaton $\mathcal{A}$ over Σ with set of accepting states F or with coloring Ω, we consider several acceptance modes:

- $L_*(\mathcal{A}) \subseteq \Sigma^*$ denotes the set of finite words over Σ accepted by $\mathcal{A}$, i.e., the set of words that have a run that ends in F.
- $L_\exists(\mathcal{A}) \subseteq \Sigma^\omega$ denotes the set of infinite words over Σ that have a run visiting an accepting state at least once, called reachability acceptance. Due to completeness, we have $L_\exists(\mathcal{A}) = L_*(\mathcal{A}) \cdot \Sigma^\omega$.
- Dually, $L_\forall(\mathcal{A}) \subseteq \Sigma^\omega$ denotes the set of infinite words over Σ that have a run which only visits accepting states, called safety acceptance.
- $L_p(\mathcal{A}) \subseteq \Sigma^\omega$ denotes the set of infinite words that have a run such that the maximal color visited infinitely often during this run is even, the classical (max)-parity acceptance.

A reachability condition is a language that is accepted by an automaton with reachability acceptance, called a reachability automaton. Safety and parity conditions and automata are defined similarly. Every (deterministic) automaton with reachability or safety acceptance can be turned into a (deterministic) parity automaton of the same size that recognizes the same language. Deterministic parity automata recognize exactly the ω-regular languages.

Recall 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 state, i.e., one that is not in F in the case of safety and one with odd color in the

case of parity, 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. As we are interested in pure reachability conditions, we impose this restriction.

Given a language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$ we denote by $\mathrm{pr}_0(L)$ its projection to the first component. Similarly, given an automaton $\mathcal{A}$ over $\Sigma_I \times \Sigma_O$, we denote by $\mathrm{pr}_0(\mathcal{A})$ the automaton obtained by projecting each letter to its first component.

Remark 1. Let $\mathrm{acc} \in \{*, \forall, \exists, p\}$, then $\mathrm{pr}_0(L_{\mathrm{acc}}(\mathcal{A})) = L_{\mathrm{acc}}(\mathrm{pr}_0(\mathcal{A}))$.

2.2 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 $i \in \mathbb{N}$ 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 and only 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 p. We say that a player wins $\Gamma_f(L)$, if she has a winning strategy.

Note that if L is recognizable by a (deterministic) parity automaton, then $\Gamma_f(L)$ is determined, i.e., exactly one of the players has a winning strategy, as delay games with parity condition can be expressed as an explicit parity game in a countable arena, which is determined [13,14].

Also, note that universality of $\mathrm{pr}_0(L)$ is a necessary condition for Player O to win $\Gamma_f(L)$. Otherwise, Player I could pick a word from $\Sigma_I^\omega \setminus \mathrm{pr}_0(L)$, which is winning for him, no matter how Player O answers.

Proposition 1. *If Player O wins $\Gamma_f(L)$, then $\mathrm{pr}_0(L)$ is universal.*

We conclude with two examples of delay games with ω-regular conditions.

Example 1. Note that both conditions can be accepted by safety automata.

1. Consider L_1 over $\{a, b, c\} \times \{b, c\}$ with $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots \in L_1$, 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.

We claim that Player I wins $\Gamma_f(L_1)$ for every 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 $\Gamma_f(L_1)$.

2. Now, consider L_2 over $\{a,b,c\} \times \{a,b,c\}$ where $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)} \cdots \in L_2$, if $\beta(n) = \alpha(n+2)$ for every $n \in \mathbb{N}$, i.e., Player O wins if the input is shifted two positions to the left.
Player O has a winning strategy for $\Gamma_f(L_2)$ for every f with $f(0) \geq 3$. In this case, Player O has at least three letters lookahead in each round, which suffices to shift the input of Player I two positions to the left. On the other hand, if $f(0) < 3$, then Player I has a winning strategy, since Player O has to pick $\beta(0)$ before $\alpha(0+2)$ has been picked by Player I.

3 Lower Bounds on the Lookahead

In this section, we prove lower bounds on the necessary lookahead for Player O to win delay games. We first give an exponential lower bound for reachability conditions, then we extend this idea to provide an exponential lower bound for safety conditions. Consequently, the same bounds hold for more expressive conditions like Büchi, co-Büchi, and parity and are complemented by an exponential upper bound for parity conditions in the next section. Note that both lower bounds hold already 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

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

First, we show that every $w \in L$ satisfies $|w| < 2^n$. To this end, we prove the stronger statement $|w| < 2^m$, where m is the maximal letter occurring in w, by induction over m. The induction base $m = 1$ is trivial, so let $m > 1$. There cannot be two occurrences of m in w, as they would constitute a bad m-pair. Accordingly, there is exactly one m in w, i.e., we can decompose w into $w = w_\triangleleft\, m\, w_\triangleright$ such that $w_\triangleleft$ and $w_\triangleright$ contain no occurrence of m. Thus, the induction hypothesis is applicable and shows $|w_\triangleleft|, |w_\triangleright| < 2^{m-1}$, which implies $|w| < 2^m$.

Dually, there is a word $w_n \in L$ with $|w_n| = 2^n - 1$ which is defined inductively via $w_1 = 1$ and $w_m = w_{m-1}\, m\, w_{m-1}$ for $m > 1$. A simple induction shows $w_n \in L$ and $|w_n| = 2^n - 1$.

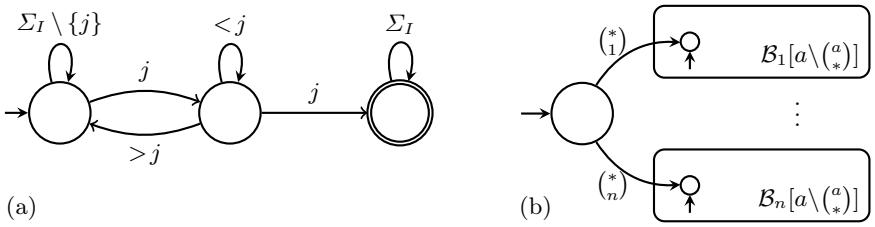

Fig. 1. (a) Automaton $\mathcal{B}_j$ for $j \in \Sigma_I$. (b) Construction of $\mathcal{A}_n$.

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's moves contain 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 constant 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. □

For safety conditions, we use the same idea as for the reachability case, but we need to introduce a new letter # to give Player I the possibility to reach a non-accepting state.

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 $f(0) \leq 2^n$.*

Proof. Let $\Sigma_I = \Sigma_O = \{1, 2, \ldots, n, \#\}$ and let w_n be defined as above. We introduce a new automaton $\mathcal{B}'_\#$ and extend every automaton $\mathcal{B}_j$ from the previous proof to $\mathcal{B}'_j$ as depicted in Figure 2. The automaton $\mathcal{A}'_n$ is constructed as in the previous proof using the automata $\mathcal{B}'_j$ and $\mathcal{B}'_\#$ instead of the $\mathcal{B}_j$.

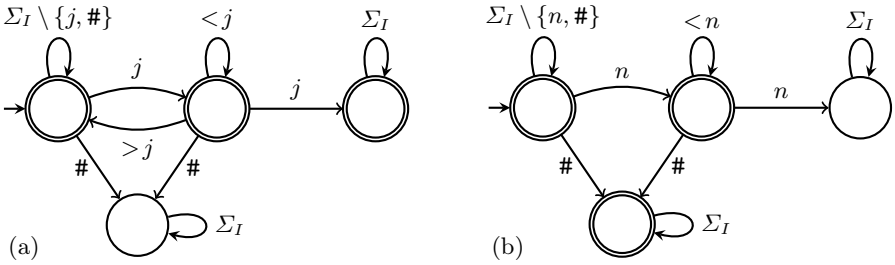

Fig. 2. (a) Automaton $\mathcal{B}'_j$ for $j \in \Sigma_I \setminus \{\#\}$. (b) Automaton $\mathcal{B}'_\#$.

For $f(0) > 2^n$, Player O wins the game: assume Player I picks u_0 in the first round and let u'_0 be u_0 without its first letter. If u'_0 contains a # preceded by at most one n, then Player O answers with # in the first round. If there is more than one n before the first # in u'_0, then she answers with n. Finally, if there is no # in u'_0, she can pick a j such that u'_0 contains a bad j-pair. All outcomes are winning for Player O.

Player I still wins the game for a constant delay function f with $f(0) \leq 2^n$ by picking the prefix of $1w_n$ of length $f(0)$ in the first round: if Player O picks some $j \in \Sigma_O \setminus \{\#\}$ in the first round, then Player I just has to answer with #. Otherwise, if Player O picks # in the first round, then Player I continues with n^ω. He wins in both situations. □

The same constructions also work with constant-size alphabets, if we encode every $j \in \{1, \ldots, n\}$ in binary with the most significant bit in the first position. Then, the natural ordering on $\{1, \ldots, n\}$ is exactly the lexicographical ordering on the corresponding bit-string representation. Accordingly, we can encode every $\mathcal{B}_j$, $\mathcal{B}'_j$, and $\mathcal{B}'_\#$ in logarithmic size in n, as deciding whether the input represents j, is larger than j, or smaller than j can be done bit-wise. Together with a binary decision tree of size $\mathcal{O}(n)$ for the initial choice of Player O we obtain deterministic 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 lower 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 of the winning condition is a necessary condition for Player O 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 universality of automata, which is PSPACE-complete. Thus, solving delay games with reachability conditions is PSPACE-complete as well. Furthermore, our proof yields an exponential upper bound on the necessary lookahead in delay games with reachability conditions.

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. *$\mathrm{pr}_0(L)$ is universal.*

Proof. The implication 2. $\Rightarrow$ 1. is trivial and 1. $\Rightarrow$ 3. is given by Proposition 1. It remains to show 3. $\Rightarrow$ 2.

We assume w.l.o.g. that the accepting states of $\mathcal{A}$ are sinks, which implies that $L_*(\mathrm{pr}_0(\mathcal{A}))$ is suffix-closed, i.e., $w \in L_*(\mathrm{pr}_0(\mathcal{A}))$ implies $ww' \in L_*(\mathrm{pr}_0(\mathcal{A}))$ for every $w' \in \Sigma_I^*$. Furthermore, let $\mathcal{A}^c$ be an automaton recognizing the complement of $L_*(\mathrm{pr}_0(\mathcal{A}))$, 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 $\alpha \in L_\exists(\mathrm{pr}_0(\mathcal{A}))$, i.e., there is a prefix of α in $L_*(\mathrm{pr}_0(\mathcal{A}))$. Thus, the prefix is in $L_*(\mathrm{pr}_0(\mathcal{A}))$ 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, $w \in L_*(\mathrm{pr}_0(\mathcal{A}))$ 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 $u_0 \in L_*(\mathrm{pr}_0(\mathcal{A}))$. 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 $\beta(0), \cdots, \beta(f(0)-1)$ in the first $f(0)$ rounds, Player O wins the play, no matter how it is continued. Hence, she has a winning strategy. □

The exponential upper bound on the necessary lookahead to win delay games with reachability conditions matches the lower bound presented in the previous section. Also, note that the upper bound holds for non-deterministic automata while the lower bound holds for deterministic automata.

Theorem 3 shows that solving delay games with reachability conditions is equivalent to universality of non-deterministic reachability automata. This problem is PSPACE-complete, which can be shown by a reduction from (to) the universality problem for finite (Büchi) automata, respectively, which are both PSPACE-complete [15,16], yielding PSPACE-completeness of solving delay games

with reachability conditions. Note that hardness holds already for deterministic automata.

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?*

Proof. To prove hardness, we proceed by a reduction from the universality problem for non-deterministic finite automata, which is PSPACE-complete [15]. We begin by turning such an automaton $\mathcal{B}$ over Σ into a non-deterministic automaton $\mathcal{B}'$ over $\Sigma \cup \{\#\}$, where $\#$ is a new letter, such that

$$L_*(\mathcal{B}) = \Sigma^* \quad \Leftrightarrow \quad L_\exists(\mathcal{B}') = (\Sigma \cup \{\#\})^\omega. \tag{1}$$

To this end, we first make all states non-accepting, then add a fresh accepting state, equipped with a self-loop labeled by every letter from Σ, and add an edge labeled by $\#$ from each formerly accepting state to the new accepting state. The resulting automaton accepts every infinite word containing exactly one $\#$ whose prefix before the $\#$ is in $L_*(\mathcal{B})$. The automaton $\mathcal{B}'$ is then obtained by a union construction with an automaton over $\Sigma \cup \{\#\}$ accepting exactly those infinite words containing no or at least two $\#$. Hence, $L_\exists(\mathcal{B}')$ is universal if and only if $L_*(\mathcal{B})$ is universal. Note that $\mathcal{B}'$ can be constructed such that $|\mathcal{B}'| \in \mathcal{O}(|\mathcal{B}|)$.

Now, we can turn $\mathcal{B}'$ into a deterministic automaton $\mathcal{A}$ by extending the alphabet, i.e., we replace each transition (p, a, q) in $\mathcal{B}'$ by $(p, \binom{a}{q}, q)$ and obtain

$$L_\exists(\pi_1(\mathcal{A})) = L_\exists(\mathcal{B}'). \tag{2}$$

Player O wins $\Gamma_f(L_\exists(\mathcal{A}))$ for some delay function f if and only if $L_*(\mathcal{B})$ is universal. This follows from the equivalence of the first and third statement of Theorem 3 and from (2) and (1). This completes the hardness proof.

To prove membership, consider a game with winning condition $L_\exists(\mathcal{A})$, where $\mathcal{A}$ is possibly non-deterministic. Due to Theorem 3, it suffices to show that universality of $\mathrm{pr}_0(L_\exists(\mathcal{A})) = L_\exists(\mathrm{pr}_0(\mathcal{A}))$ is decidable in PSPACE. By doubling the state space of $\mathrm{pr}_0(\mathcal{A})$ we can turn it into into a Büchi automaton $\mathcal{A}'$ accepting the same language. The universality problem for Büchi automata is in PSPACE [16], which concludes the proof. □

Another consequence of the proof of Theorem 3 concerns the strategy complexity of delay games with reachability conditions: if Player O wins for some delay function, then she has a winning strategy that receives exponentially many input letters and answers by also giving exponentially many output letters and thereby already guarantees a winning play, i.e., all later moves are irrelevant. Thus, the situation is similar to classical reachability games on graphs, in which positional attractor strategies allow a player to guarantee a win after a bounded number of moves. The strategy described above can be implemented by a lookup table that maps all minimal words in $L_*(\mathrm{pr}_0(\mathcal{A}))$ to a word in Σ_O^* of the same length such that the combined word is accepted by $\mathcal{A}$.

4.2 Safety Conditions

Unsurprisingly, Example 1.1 shows that Theorem 4.1 does not hold for safety conditions: the projection $\mathrm{pr}_0(L_1)$ is universal, but Payer 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): we show solving delay games with safety conditions to be ExpTime-hard by a reduction from the acceptance problem for alternating polynomial space Turing machines [11].

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?*

Proof. Let $\mathcal{M} = (Q, Q_\exists, Q_\forall, \Sigma, q_I, \Delta, F)$ be an alternating polynomial space Turing machine, where $\Delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{-1, 0, 1\}$ is the transition relation, and let $x \in \Sigma^*$ be an input. For technical reasons we assume the accepting states in F to be equipped with a self-loop. Furthermore, let p be a polynomial that bounds $\mathcal{M}$'s space consumption. We construct a safety automaton $\mathcal{A}$ of polynomial size in $|\Delta|$ and $p(|x|)$ such that $\mathcal{M}$ rejects x if and only if Player O wins $\Gamma_f(L_\forall(\mathcal{A}))$ for some f. This is sufficient, since APSpace = ExpTime is closed under complements. Thus, we give Player I control over the existential transitions while Player O controls the universal ones, but Player I is in charge of producing the transitions with his moves. He can copy configurations in order to wait for Player O's choices for the universal configurations, which are delayed due to the lookahead.

Formally, the input alphabet Σ_I contains $\Sigma \cup Q$ and two separators N and C while the output alphabet Σ_O contains Δ and two signals ✗ and ✓. Intuitively, Player I produces configurations of $\mathcal{M}$ of length $p(|x|)$ preceded by either N or C 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 letter from Δ. At every other position, she has to pick one of the signals: ✗ 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 $\mathcal{A}$ is the product of safety automata checking the following properties of an input word $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots \in (\Sigma_I \times \Sigma_O)^\omega$:

1. $\alpha \in (\{N, C\} \cdot \mathrm{Conf})^\omega$, where Conf is the set of encodings of configurations of length $p(|x|)$, i.e., words of length $p(|x|)+1$ over $\Sigma \cup Q$ that contain exactly one letter from Q. If this is not the case, then the product automaton $\mathcal{A}$ goes to an accepting sink, i.e., in order to win, Player I has to produce an α that satisfies the requirement.
2. $\beta \in (\Delta \cdot \{✓, ✗\}^{p(|x|+1)})^\omega$. If this is not the case, then the product automaton $\mathcal{A}$ goes to an rejecting sink, i.e., in order to win, Player O has to produce a β that satisfies the requirement.

[1] If the following configuration is existential, then her choice is ignored.

3. The first configuration picked by Player I is the initial one of $\mathcal{M}$ on x. If this is not the case, $\mathcal{A}$ goes to an accepting sink.
4. If β contains a ✗, then the automaton checks whether there is indeed an error by doing the following at the first occurrence of ✗: it stores the previous, the current, and the next input letter, the transition picked by Player O at the last separator N, and whether the current configuration is existential or universal. Some of this information has to be stored continuously, since these letters appear before the first ✗. This is possible using a set of states whose size is polynomial in $|\Sigma \cup Q \cup \Sigma|$.
   Then, the automaton processes $p(|x|) + 1$ letters (and remembers whether it traverses the separator N or C), and then checks whether the letter just reached is updated correctly or not:
   - If the separator is C, then the current letter is updated correctly, if it is equal to the marked one.
   - If the separator is N and the configuration in which the error was marked is existential, then the letter is updated correctly, if there is a transition of $\mathcal{M}$ that is compatible with the current letter and the marked one.
   - If the separator is N and the configuration in which the error was marked is universal, then the letter is updated correctly, if it is compatible with the transition picked by Player O at the last separator before the ✗. If she has picked a transition that is not applicable to the current configuration, $\mathcal{A}$ goes to an accepting sink.

   If the update is not correct, i.e., Player O has correctly claimed an error, then $\mathcal{A}$ goes to an accepting sink. Otherwise, it goes to an rejecting sink, i.e., in order to win, Player O should only claim an error at an incorrect update of a configuration, but she wins if she claims an error correctly. All subsequent claims by Player O are ignored, i.e., after the first claim is evaluated, the play is either accepting or rejecting, no matter how it is continued.
5. Finally, if α contains an accepting state, then $\mathcal{A}$ goes to a rejecting sink, unless Player O correctly claims an error in the preceding configuration.

All these properties can be checked by safety automata whose size is polynomial in the size of $\mathcal{M}$ and in $p(|x|)$. All non-sink states of $\mathcal{A}$ are accepting, i.e., as long as both players stick to their requirements, Player I starts with the initial configuration, does not introduce an error, and never reaches a terminal configuration, Player O does not incorrectly claim an error, and no accepting state is reached, then the input is accepted.

It remains to prove that $\mathcal{M}$ rejects x if and only if Player O wins $\Gamma_f(L_\forall(\mathcal{A}))$ for some f.

"$\Rightarrow$": Assume $\mathcal{M}$ rejects x and let f be the constant delay function with $f(0) = p(|x|) + 2$. We show that Player O wins $\Gamma_f(L_\forall(\mathcal{A}))$. To this end, consider a play of $\Gamma_f(L_\forall(\mathcal{A}))$. At every time, Player O has enough lookahead to correctly claim the first error introduced by Player I, if he introduces one. Furthermore, she has access to the whole encoding of each universal configuration, whose successor she has to determine. This allows her to simulate the rejecting run of $\mathcal{M}$ on x, which does not reach an accepting state, no matter which transitions

Player I picks. Thus, he has to introduce an error in order to win, which Player O can detect using the lookahead. If Player I does not introduce an error, the play proceeds either ad infinitum or until he reaches a terminal state, at which he has to introduce an error to continue. In every case, $\mathcal{A}$ accepts the resulting play, i.e., Player O wins. Thus, Player O has a winning strategy for $\Gamma_f(L_\forall(\mathcal{A}))$.

"$\Leftarrow$": We show the contrapositive. Assume that $\mathcal{M}$ accepts x and let f be an arbitrary delay function. We show that Player O wins $\Gamma_f(L_\forall(\mathcal{A}))$. To this end, consider a play of $\Gamma_f(L_\forall(\mathcal{A}))$. Player I starts with the initial configuration and picks the successor configuration of an existential one according to the accepting run, and copies universal configurations as often as necessary to obtain a play prefix in which Player O has to determine the transition she wants to apply in this configuration. Thus, he will eventually produce an accepting configuration of $\mathcal{M}$ without ever introducing an error. Hence, either Player O incorrectly claims an error or the play reaches an accepting state. In either case, Player I wins the resulting play, i.e., he has a winning strategy for $\Gamma_f(L_\forall(\mathcal{A}))$. □

It is noteworthy that the lower bound just proven does not require the full exponential lookahead that is necessary in some cases and sufficient in all cases to win a delay game with safety conditions: Player O wins the game constructed above with sublinear lookahead, as $p(|x|)+2$ is smaller than the size of $\mathcal{A}$. Thus, determining the winner of a delay game with safety condition w.r.t. linearly bounded delay is already ExpTime-hard.

4.3 Parity Conditions

In the previous two subsections, we showed solving delay games with reachability conditions to be PSpace-complete and solving games with safety conditions to be ExpTime-hard. To conclude this section, we complement the latter lower bound 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 of the form $2^{(nk)^2}$ on the necessary lookahead from the algorithm, where n is the size and k the number of colors of the automaton. To conclude this section, we lower the upper bound to 2^{nk} via a direct pumping argument, which matches the lower bound from the previous section. Note that all results only hold for deterministic automata.

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

To prove this result, we construct 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 $\Omega\colon Q \to \mathbb{N}$ and let $n = |Q \times \Omega(Q)|$. First, we adapt $\mathcal{A}$ to keep track of the maximal color visited during a run. Formally, we define $\mathcal{C} = (Q_\mathcal{C}, \Sigma_I \times \Sigma_O, q_I^\mathcal{C}, \delta_\mathcal{C}, \Omega_\mathcal{C})$ of size n 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
– $\Omega_{\mathcal{C}}(q,c) = c$.

Note that $\mathcal{C}$ does not recognize $L_p(\mathcal{A})$. However, we are not interested in complete runs of $\mathcal{C}$, but only 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 $c_{|w|} = \max\{\Omega(q_j) \mid 0 \leq j \leq |w|\}$.

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 which states Player O can enforce by picking an appropriate completion, which are 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 $\mathrm{pr}_0(\mathcal{C})$. 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, 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. Now, we define $\mathfrak{R} = \{r \mid \mathrm{dom}(r) \neq \emptyset \text{ and } W_r \text{ is infinite}\}$.

Lemma 1. *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 $r \neq r' \in \mathfrak{R}$ such that $\mathrm{dom}(r) = \mathrm{dom}(r')$. Then, $W_r \cap W_{r'} = \emptyset$.*
4. *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 $w \in W_r$.*

Due to items 3. and 4. of Lemma 1, we can define for every non-empty $D \subseteq Q_C$ a function r_D that maps words w with $|w| \geq 2^{n^2}$ to the unique function r with $\mathrm{dom}(r) = D$ and $w \in W_r$.

Now, we can define the 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 such that

$$\mathrm{dom}(r_0) = \{q_I^{\mathcal{C}}\} \tag{C1}$$

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 such that

$$\mathrm{dom}(r_i) = r_{i-1}(q_{i-1}). \tag{C2}$$

Then, Player O picks some state $q_i \in \mathrm{dom}(r_i)$. Both players can always move: Player I as $r_{i-1}(q_{i-1})$ is always non-empty (Lemma 1.1) and thus the domain of some $r \in \mathfrak{R}$ (Lemma 1.4), Player O 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 $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$, which 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 p. As usual, we say that a player wins $\mathcal{G}(\mathcal{A})$, if she has a winning strategy.

Lemma 2. *Player O wins $\Gamma_f(L_p(\mathcal{A}))$ for some delay function f if and only if Player O wins $\mathcal{G}(\mathcal{A})$.*

Proof. For the sake of readability, we denote $\mathcal{G}(\mathcal{A})$ by $\mathcal{G}$ and $\Gamma_f(L_p(\mathcal{A}))$ by Γ.

"⇒": Let τ_O be a winning strategy for Player O in Γ where we assume[2] f to be constant [5]. We construct a winning strategy τ_O' for Player O in $\mathcal{G}$ by simulating a play of $\mathcal{G}$ by a play of Γ.

Let r_0 be the first move of Player I in $\mathcal{G}$, which has to be answered by Player O by picking $\tau_O'(r_0) = q_I^{\mathcal{C}}$, and let r_1 be Player I's response. As W_{r_0} and W_{r_1} are infinite by definition, we can choose witnesses $w_0 \in W_{r_0}$ and $w_1 \in W_{r_1}$ such that $|w_1| \geq f(0)$. We simulate this play prefix in Γ: Player I picks $w_0 w_1 = \alpha(0) \cdots \alpha(\ell_1 - 1)$ in his first moves and let $\beta(0) \cdots \beta(\ell_1 - f(0))$ be the response of Player O according to τ_O. Then, we obtain $|\beta(0) \cdots \beta(\ell_1 - f(0))| \geq |w_0|$, as $|w_1| \geq f(0)$. Thus, we are in the following situation for $i = 1$:

[2] This assumption simplifies the proof, but can be avoided at the expense of introducing intricate notation.

– in $\mathcal{G}$, we have constructed a play prefix $r_0q_0 \cdots r_{i-1}q_{i-1}r_i$,
– in Γ, Player I has picked $w_0 \cdots w_i = \alpha(0) \cdots \alpha(\ell_i - 1)$ and Player O has picked $\beta(0) \cdots \beta(\ell_i - f(0))$ according to τ_O. The moves of the players satisfy $|\beta(0) \cdots \beta(\ell_i - f(0))| \geq |w_0 \cdots w_{i-1}|$.
– Finally, w_j is a witness for r_j for every $j \leq i$.

Now, let $i \geq 1$ be arbitrary and let $q_{i-1} = (q'_{i-1}, c_{i-1})$. We define q_i to be the state of $\mathcal{C}$ that is reached from $(q'_{i-1}, \Omega(q'_{i-1}))$ after processing w_{i-1} and the corresponding moves of Player O, i.e.,

$$\binom{\alpha(|w_0 \cdots w_{i-2}|)}{\beta(|w_0 \cdots w_{i-2}|)} \cdots \binom{\alpha(|w_0 \cdots w_{i-1}| - 1)}{\beta(|w_0 \cdots w_{i-1}| - 1)}.$$

By definition of r_{i-1}, we have $q_i \in r_{i-1}(q_{i-1})$. Correspondingly, we can define $\tau'_O(r_0q_0 \cdots r_{i-1}q_{i-1}r_i) = q_i$. Now, let r_{i+1} be the next move of Player I in $\mathcal{G}$ and let $w_{i+1} \in W_{r_{i+1}}$ be a witness with $|w_{i+1}| \geq f(0)$. In Γ, let Player I pick $w_{i+1} = \alpha(\ell_i) \cdots \alpha(\ell_{i+1} - 1)$ as his next moves and let Player O respond by $\beta(\ell_i - f(0) + 1) \cdots \beta(\ell_{i+1} - f(0))$ according to τ_O. Thus, we are again in the aforementioned situation for $i + 1$, which concludes the definition of τ'_O.

It remains to show that τ'_O is winning. Consider a play $r_0q_0r_1q_1r_2q_2 \cdots$ that is consistent with τ'_O and let $w = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)} \cdots$ be the outcome in Γ constructed during the simulation as defined above. Note that $\alpha(0)\alpha(1)\alpha(2) \cdots$ is equal to $w_0w_1w_2 \cdots$, where each w_i is a witness of r_i. Furthermore, let $q_i = (q'_i, c_i)$.

A straightforward inductive application of Remark 2 shows that q'_{i+1} is the state that $\mathcal{A}$ reaches after processing w_i and the corresponding moves of Player O starting in q'_i and that c_{i+1} is the maximal color seen on the run. Thus, the maximal color visited infinitely often by $\mathcal{A}$ after processing w is the same as the maximal color of the sequence $c_0c_1c_2 \cdots$, which is even, as w is consistent with a winning strategy and therefore accepted by $\mathcal{A}$. Hence, $r_0q_0r_1q_1r_2q_2 \cdots$ is winning for Player O and τ'_O is a winning strategy.

"$\Leftarrow$": Let τ'_O be a wining strategy for Player O in $\mathcal{G}$. We construct a winning strategy τ_O for her in Γ for the constant delay function f with $f(0) = 2d$, where $d = 2^{n^2}$. The strategy τ_O is again constructed by simulating a play of Γ by a play of $\mathcal{G}$.

In the following, both players pick their moves in blocks of length d. We denote Player I's blocks by $\overline{a_i}$ and Player O's blocks by $\overline{b_i}$, i.e., every $\overline{a_i}$ is in Σ_I^d and every $\overline{b_i}$ is in Σ_O^d.

Let $\overline{a_0a_1}$ be the first move of Player I in Γ, define $q_0 = q_I^{\mathcal{C}}$, and let $r_0 = r_{\{q_0\}}(\overline{a_0})$ and $r_1 = r_{r_0(q_0)}(\overline{a_1})$. Then, $r_0q_0r_1$ is a play prefix in $\mathcal{G}$ that is consistent with τ'_O. Thus, we are in the following situation for $i = 1$:

– in Γ, Player I has picked blocks $\overline{a_0} \cdots \overline{a_i}$ and Player O has picked $\overline{b_0} \cdots \overline{b_{i-2}}$,
– in $\mathcal{G}$ we have constructed a play prefix $r_0q_0 \cdots r_{i-1}q_{i-1}r_i$ that is consistent with τ'_O, and
– $\overline{a_j}$ is a witness for r_j for every $j \leq i$.

Now, let $i \geq 1$ be arbitrary and $q_i = \tau'_O(r_0 q_0 \cdots r_{i-1} q_{i-1} r_i)$. The rules of $\mathcal{G}$ imply $q_i \in \operatorname{dom}(r_i) = r_{i-1}(q_{i-1})$. Furthermore, as $\overline{a_{i-1}}$ is a witness for r_{i-1}, there is some $\overline{b_{i-1}}$ such that the automaton $\mathcal{C}$ reaches q_i after processing $\binom{\overline{a_{i-1}}}{\overline{b_{i-1}}}$ from $(q'_{i-1}, \Omega(q'_{i-1}))$, where $q_{i-1} = (q'_{i-1}, c_{i-1})$. Player O's strategy for Γ is to play the letters of $\overline{b_{i-1}}$ in the next d rounds. These are answered by Player I by d letters forming $\overline{a_{i+1}}$. This way, we obtain $r_{i+1} = r_{r_i(q_i)}(\overline{a_{i+1}})$ bringing us back to the aforementioned situation for $i+1$, which concludes the definition of τ_O.

It remains to show that τ_O is winning for Player O. Let $w = \binom{\overline{a_0}}{\overline{b_0}}\binom{\overline{a_1}}{\overline{b_1}}\binom{\overline{a_2}}{\overline{b_2}}\cdots$ be the outcome of a play of Γ consistent with τ_O. Also, let $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$ be the corresponding play of $\mathcal{G}$ constructed as described in the simulation above, where each $\overline{a_i}$ is a witness for r_i. Let $q_i = (q'_i, c_i)$.

A straightforward inductive application of Remark 2 shows that q'_{i+1} is the state reached in $\mathcal{A}$ after processing $\binom{\overline{a_i}}{\overline{b_i}}$ starting in q'_i and that c_{i+1} is the largest color seen on this run. As $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$ is consistent with τ'_O, the sequence $\Omega(q_0)\Omega(q_1)\Omega(q_2)\cdots = c_0 c_1 c_2 \cdots$ satisfies the parity condition, i.e., the maximal color occurring infinitely often is even. Thus, the maximal color occurring infinitely often during the run of $\mathcal{A}$ on w is even as well, i.e., w is winning for Player O. Thus, τ_O is a winning strategy for Player O. □

Now, we can prove the main theorem of this section.

Proof (Theorem 5). Due to Lemma 2, we just have to show that we can construct and solve an explicit version of $\mathcal{G}(\mathcal{A})$ in exponential time.

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 Lemma 1.2 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.

Now, we can encode $\mathcal{G}(\mathcal{A})$ as a graph-based parity game[3] $((V, V_I, V_O, E), \Omega')$ with $\Omega' \colon V \to \mathbb{N}$ where

- $V = V_I \cup V_O$,
- $V_I = \{v_I\} \cup \mathfrak{R} \times Q_\mathcal{C}$ for some fresh initial vertex v_I,
- $V_O = \mathfrak{R}$,
- E is the union of the following sets of edges:
  - $\{(v_I, r) \mid \operatorname{dom}(r) = \{q_I^\mathcal{C}\}\}$: the initial moves of Player I.
  - $\{((r,q), r') \mid \operatorname{dom}(r') = r(q)\}$: (regular) moves of Player I.
  - $\{(r, (r,q)) \mid q \in \operatorname{dom}(r)\}$: moves of Player O, and
- $\Omega'(v) = \begin{cases} c & \text{if } v = (r,(q,c)) \in \mathfrak{R} \times Q_\mathcal{C}, \\ 0 & \text{otherwise.} \end{cases}$

Then, Player O wins $\mathcal{G}(\mathcal{A})$ if and only if she has a winning strategy from v_I in the explicit parity game. As a parity game with n vertices, m edges, and k colors can be solved in time $\mathcal{O}(mn^{\frac{k}{3}})$ [18] the latter problem can be solved in exponential time in the size of $\mathcal{A}$. □

[3] For a complete definition see, e.g., [17]

We obtain an exponential upper bound of the form $2^{(|\mathcal{A}|k)^2+1}$ on the delay necessary for Player O to win a delay game with parity condition by applying both directions of the equivalence between $\Gamma_f(L_p(\mathcal{A}))$ and $\mathcal{G}(\mathcal{A})$. The square in the exponent can be avoided by a direct pumping argument which yields a stronger bound.

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 $f(0) \leq 2^{2|A|k+2} + 2$.*

Proof. We only consider the non-trivial implication 1. $\Rightarrow$ 2. We define $f \leq f'$ if $f(i) \leq f'(i)$ for every i, which boils down to $f(0) \leq f'(0)$ for constant delay functions. If $f \leq f'$ and Player I wins $\Gamma_{f'}(L)$, then he wins $\Gamma_f(L)$ as well. Applying the monotonicity, determinacy, and the fact that constant delay functions are sufficient for Player O to win ω-regular delay games [5] imply that the contrapositive $\neg 2. \Rightarrow \neg 1.$ is equivalent to the following statement, which we prove below: if Player I wins the game $\Gamma_f(L)$ for the constant delay function f with $f(0) = 2^{2|A|k+2} + 2$, then also for every constant delay function $f' \geq f$.

Let $\mathcal{C} = (Q_\mathcal{C}, \Sigma_I \times \Sigma_O, q_I^\mathcal{C}, \delta_\mathcal{C}, \Omega_\mathcal{C})$ be the color-tracking extension of $\mathcal{A}$ as defined above. We extend $\mathcal{C}$ again to reset and *output* the stored color at positions determined by a marking of the letters from Σ_I. All other states have a trivial color that is irrelevant, as long as infinitely many letters of the input are marked. Formally, we define $\mathcal{D} = (Q_\mathcal{D}, \Sigma_I^\mathcal{D} \times \Sigma_O, q_I^\mathcal{D}, \delta_\mathcal{D}, \Omega_\mathcal{D})$ with

- $Q_\mathcal{D} = Q_\mathcal{C} \times \{0,1\}$
- $\Sigma_I^\mathcal{D} = \Sigma_I \times \{0,1\}$
- $q_I^\mathcal{D} = (q_I^\mathcal{C}, 0)$,
- $\delta_\mathcal{D}((q,t), \binom{(a,t')}{b}) = \begin{cases} ((q',c),t') & \text{if } t = 0, \\ ((q',\Omega(q')),t') & \text{if } t = 1, \end{cases}$ where $(q',c) = \delta_\mathcal{C}(q, \binom{a}{b})$, and
- $\Omega_\mathcal{D}((q,c),t) = t \cdot c$.

We call a letter of the form $(a,1) \in \Sigma_I^\mathcal{D}$ a trigger.

Let $w = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)} \cdots \in (\Sigma_I^\mathcal{D} \times \Sigma_O)^\omega$. Then, $w \notin L_p(\mathcal{D})$ if and only if $\binom{\mathrm{pr}_0(\alpha(0))}{\beta(0)}\binom{\mathrm{pr}_0(\alpha(1))}{\beta(1)}\binom{\mathrm{pr}_0(\alpha(2))}{\beta(2)} \cdots \notin L$ and $\alpha(0)\alpha(1)\alpha(2)\cdots$ contains infinitely many triggers. Using this, one can easily show that Player I wins $\Gamma_f(L)$ if and only if he wins $\Gamma_f(L_p(\mathcal{D}))$. For the left-to-right direction, Player I just has to introduce infinitely many triggers at arbitrary positions and for the right-to-left direction, Player I projects his moves to Σ_I. Thus, it suffices to show the following property: if Player I wins $\Gamma_f(L_p(\mathcal{D}))$ for the constant delay function f with $f(0) = 2^{2|A|k+2} + 2$, then also for every constant delay function $f' \geq f$.

To this end, let τ_I be a winning strategy for Player I for $\Gamma_f(L_p(\mathcal{D}))$, let $f' \geq f$ be arbitrary, and define $d = 2^{|\mathcal{D}|} + 1$, i.e., we have $f(0) = 2d$. We can assume w.l.o.g. that τ_I only allows outcomes that have a trigger exactly at the

positions $d-1, 2d-1, 3d-1, \ldots$. We construct a winning strategy τ_I' for Player I in $\Gamma_{f'}(L_p(\mathcal{D}))$ by simulating a play of $\Gamma_{f'}(L_p(\mathcal{D}))$ via a play of $\Gamma_f(L_p(\mathcal{D}))$. In the following, we will group the moves of the players in $\Gamma_f(L_p(\mathcal{D}))$ into blocks of length d. We denote Player I's blocks by $\overline{a_i}$ and Player O's blocks by $\overline{b_i}$. For the sake of readability, we denote $\Gamma_f(L_p(\mathcal{D}))$ by Γ and $\Gamma_{f'}(L_p(\mathcal{D}))$ by Γ'.

To begin, we have to give the first moves of Player I in Γ'. To this end, consider the first move $\tau_I(\varepsilon) = \overline{a_0 a_1}$ of Player I in Γ. By the choice of d, we can decompose $\overline{a_0}$ into $x_0 y_0 z_0$ and $\overline{a_1}$ into $x_1 y_1 z_1$ with non-empty y_0, z_0, y_1, and z_1 such that the powerset automaton of $\mathrm{pr}_0(\mathcal{D})$ reaches the same state after processing x_0 and $x_0 y_0$ and reaches the same state after processing $\overline{a_0} x_1$ and $\overline{a_0} x_1 y_1$, i.e., we have a state repetition in each block[4]. Note that z_i being non-empty implies that x_i and y_i contain no triggers.

Now, we pump y_0 sufficiently often, i.e., let $\alpha(0) \cdots \alpha(\ell_0 - 1) = x_0 (y_0)^{h_0} z_0$ for the smallest $h_0 \geq 1$ such that $|x_0 (y_0)^{h_0} z_0| \geq f'(0)$. Similarly, we define $\alpha(\ell_0) \cdots \alpha(\ell_1 - 1) = x_1 (y_1)^{h_1} z_1$ for the smallest $h_1 \geq 1$ with $|x_1 (y_1)^{h_1} z_1| \geq f'(0)$.

Now, we define τ_I' such that Player I picks $\alpha(0) \cdots \alpha(\ell_1 - 1)$ with his first moves in Γ', which is answered by Player O by picking $\beta(0) \cdots \beta(\ell_1 - f'(0))$. By the choice of h_1, we obtain $(\ell_1 - f'(0)) + 1 \geq \ell_0$.

Now, we are in the following situation for $i = 1$:

- In Γ', Player I has picked $\alpha(0) \cdots \alpha(\ell_i - 1)$ such that for every $j \leq i$: $\alpha(\ell_{j-1}) \cdots \alpha(\ell_j - 1) = x_j (y_j)^{h_j} z_j$.
- In Γ', Player O has picked $\beta(0) \cdots \beta(\ell_i - f'(0))$ such that $(\ell_i - f'(0)) + 1 \geq \ell_{i-1}$. Thus, Player O has provided an answer to $\alpha(\ell_{i-2}) \cdots \alpha(\ell_{i-1} - 1) = x_{i-1} (y_{i-1})^{h_{i-1}} z_{i-1}$.
- In Γ, Player I has picked blocks $\overline{a_0} \cdots \overline{a_i}$ and Player O has picked $\overline{b_0} \cdots \overline{b_{i-2}}$.

Now, let $i \geq 1$ be arbitrary and let q_{i-1} be the state reached by $\mathcal{D}$ after processing $\binom{\overline{a_0}}{\overline{b_0}} \cdots \binom{\overline{a_{i-2}}}{\overline{b_{i-2}}}$. Furthermore, let q_{i-1}^* be the state is reached by $\mathcal{D}$ after processing $x_{i-1} (y_{i-1})^{h_{i-1}-1}$ and the corresponding letters picked by Player O at these positions starting in q_{i-1}. Due to the state repetition induced by the decomposition, we can find a word $x_{i-1}' \in \Sigma_O^{|x_{i-1}|}$ such that $\mathcal{A}$ reaches the same state q_{i-1}^* after processing $\binom{x_{i-1}}{x_{i-1}'}$ starting in q_{i-1}. We define $\overline{b_{i-1}} = x_{i-1}' y_{i-1}' z_{i-1}'$, where y_{i-1}' and z_{i-1}' are the letters picked by Player O at the positions of the last repetition of y_{i-1} and at the positions of z_{i-1}, respectively.

We continue the simulation in Γ by letting Player O pick the block $\overline{b_{i-1}}$ during the next d rounds, which is answered by Player I by picking the next block $\overline{a_{i+1}}$. Again, we can decompose $\overline{a_{i+1}}$ into $x_{i+1} y_{i+1} z_{i+1}$ with non-empty y_{i+1} and z_{i+1} such that the decomposition induces a state repetition of the powerset automaton of $\mathrm{pr}_0(\mathcal{D})$ after processing the concatenation of the previous blocks and these prefixes. As before, we define $\alpha(\ell_i) \cdots \alpha(\ell_{i+1} - 1) = x_{i+1} (y_{i+1})^{h_{i+1}} z_{i+1}$ for the smallest $h_{i+1} \geq 1$ such that $x_{i+1} (y_{i+1})^{h_{i+1}} z_{i+1} \geq f'(0)$. Now, we define τ_I' such that Player I picks $\alpha(\ell_i) \cdots \alpha(\ell_{i+1} - 1)$ with his next moves in Γ', which

[4] Note that $d = 2^{|\mathcal{D}|} + 1$ is sufficient to obtain a state repetition with non-empty y_i and z_i, since the empty set is not reachable in the powerset automaton, as $\mathcal{D}$ is complete.

is answered by Player O by picking $\beta((\ell_i - f'(0)) + 1) \cdots \beta(\ell_{i+1} - f'(0))$. By the choice of h_{i+1}, we obtain $(\ell_{i+1} - f'(0)) + 1 \geq \ell_i$. Thus, we are in the situation described above for $i + 1$, which completes the definition of τ_I'.

It remains to show that τ_I' is winning. Let $w' = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)} \cdots$ be an outcome of a play that is consistent with τ_I' in Γ' and let $w = \binom{\overline{a_0}}{b_0}\binom{\overline{a_1}}{b_1}\binom{\overline{a_2}}{b_2} \cdots$ be the simulating play of Γ as described above.

A simple induction shows that $\mathcal{D}$ reaches the same state after processing

$$\binom{\alpha(0)}{\beta(0)} \cdots \binom{\alpha(\ell_i - 1)}{\beta(\ell_i - 1)} \quad \text{and} \quad \binom{\overline{a_0}}{b_0} \cdots \binom{\overline{a_i}}{b_i}. \tag{3}$$

As the only triggers occurring in w' and in w occur as last letters of prefixes as in (3), and as there are infinitely many, acceptance of w' and w only depends on states reached after processing such a prefix. Hence, w' is accepted by $\mathcal{D}$ if and only if w is. Thus, both are not accepted, as w is consistent with the winning strategy τ_I, i.e., τ_I' is indeed a winning strategy. □

5 Conclusion

We gave the first algorithm that solves ω-regular delay games in exponential time, 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.

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. These results are presented in the appendix. In future work, we aim to find lower bounds.

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

References

1. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Trans. Amer. Math. Soc. **138** (1969) pp. 295–311
2. 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. Volume 803 of LNCS., Springer (1993) 583–621
3. 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)
4. Hosch, F.A., Landweber, L.H.: Finite delay solutions for sequential conditions. In: ICALP 1972. (1972) 45–60
5. Holtmann, M., Kaiser, L., Thomas, W.: Degrees of lookahead in regular infinite games. LMCS **8**(3) (2012)
6. Fridman, W., Löding, C., Zimmermann, M.: Degrees of Lookahead in Context-free Infinite Games. In Bezem, M., ed.: CSL 2011. Volume 12 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2011) 264–276
7. Zimmermann, M.: Delay games with WMSO+U winning conditions (2014) Under submission.
8. Carayol, A., Löding, C.: Uniformization in automata theory. In Schroeder-Heister, P., Heinzmann, G., Hodges, W., Bour, P.E., eds.: Proceedings of the Fourteenth International Congress of Logic, Methodology and Philosophy of Science, College Publications, London (2012) To appear.
9. Löding, C., Winter, S.: Synthesis of deterministic top-down tree transducers from automatic tree relations. In Peron, A., Piazza, C., eds.: GandALF 2014. Volume 161 of EPTCS. (2014) 88–101
10. Carayol, A., Löding, C.: MSO on the infinite binary tree: Choice and order. In Duparc, J., Henzinger, T.A., eds.: CSL 2007. Volume 4646 of LNCS., Springer (2007) 161–176
11. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. J. ACM **28**(1) (1981) 114–133
12. Hutagalung, M., Lange, M., Lozes, É.: Buffered simulation games for Büchi automata. In Ésik, Z., Fülöp, Z., eds.: AFL 2014. Volume 151 of EPTCS. (2014) 286–300
13. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: FOCS 1991, IEEE (1991) 368–377
14. Mostowski, A.: Games with forbidden positions. Technical Report 78, University of Gdańsk (1991)
15. Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential space. In: SWAT 1972, IEEE Computer Society (1972) 125–129
16. Sistla, A.P., Vardi, M.Y., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic (extended abstract). In Brauer, W., ed.: ICALP 1985. Volume 194 of LNCS., Springer (1985) 465–474
17. Grädel, E., Thomas, W., Wilke, T., eds.: Automata, Logics, and Infinite Games: A Guide to Current Research. Volume 2500 of LNCS., Springer (2002)
18. Schewe, S.: Solving parity games in big steps. In Arvind, V., Prasad, S., eds.: FSTTCS 2007. Volume 4855 of LNCS., Springer (2007) 449–460

A Appendix

A.1 Winning Conditions that are Safety and Reachability

The lower bounds on the necessary lookahead for reachability and safety conditions presented in Section 3 rely on the same construction. One can even turn the safety automata $\mathcal{A}'_n$ exhibiting the lower bound for safety conditions into reachability automata exhibiting the same lower bounds for reachability conditions by changing the set of accepting states, but leaving the transition structure of $\mathcal{A}'_n$ unchanged. Nevertheless, these automata will not accept the same language. In the following, we show that this is inevitable: polynomial lookahead suffices for winning conditions that are both reachability and safety. Furthermore, solving such games is in the second level of the polynomial hierarchy.

Our results follow from the proof of the following well-known lemma that characterizes languages that are both reachability and safety conditions.

Lemma 3. *Let $L = L_\exists(\mathcal{A}_R) = L_\forall(\mathcal{A}_S)$ for a deterministic automaton $\mathcal{A}_R$ and a non-deterministic automaton $\mathcal{A}_S$ over $\Sigma_I \times \Sigma_O$. Then, $L = K \cdot (\Sigma_I \times \Sigma_O)^\omega$ for some $K \subseteq (\Sigma_I \times \Sigma_O)^{\leq k}$ with $k \leq |\mathcal{A}_R| \cdot |\mathcal{A}_S|$.*

Proof. Fix $k = |\mathcal{A}_R| \cdot |\mathcal{A}_S|$. It suffices to show that every $\alpha \in L$ has a prefix of length at most k such that $\mathcal{A}_R$ reaches an accepting state after processing the prefix. Then, $K = L_*(\mathcal{A}_R) \cap (\Sigma_I \times \Sigma_O)^{\leq k}$ has the desired properties.

Towards a contradiction, assume $\alpha \in L$ does not have such a prefix and fix accepting runs ρ_R of $\mathcal{A}_R$ and ρ_S of $\mathcal{A}_S$ on α. There are positions i, j with $0 \leq i < j \leq k$ such that both runs are in the same state at positions i and j. Furthermore, the prefix of ρ_R up to position j does not contain an accepting state and the prefix of ρ_S up to position j only contains accepting states.

On the one hand, this implies that $\alpha' = \alpha(0) \cdots \alpha(i)(\alpha(i+1) \cdots \alpha(j))^\omega$ is not accepted by $\mathcal{A}_R$, as the unique run on it only visits states also visited by the prefix of ρ_R up to position j. On the other hand, the same word has an accepting run of $\mathcal{A}_S$, as the loop of the prefix of ρ_S up to position j can be repeated forever. This contradicts $L_\forall(\mathcal{A}_R) = L_\exists(\mathcal{A}_S)$. □

Our main theorem shows that the bound $|\mathcal{A}_R| \cdot |\mathcal{A}_S|$ is sufficient for Player O to win the delay game, if she wins it at all.

Theorem 7. *Let $L = L_\forall(\mathcal{A}_R) = L_\exists(\mathcal{A}_S)$ for a deterministic automaton $\mathcal{A}_R$ and a non-deterministic automaton $\mathcal{A}_S$ over $(\Sigma_I \times \Sigma_O)$, and let $k = |\mathcal{A}_R| \cdot |\mathcal{A}_S|$. 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 $f(0) \leq k$.*

Proof. One implication is trivial, so assume Player O wins $\Gamma_f(L)$ for some f. Let $K \subseteq (\Sigma_I \times \Sigma_O)^{\leq k}$ be such that $L = K \cdot (\Sigma_I \times \Sigma_O)^\omega$ and define

$$K' = \{w \in (\Sigma_I \times \Sigma_O)^k \mid w \text{ has a prefix in } K\}.$$

Then, we have $K \cdot (\Sigma_I \times \Sigma_O)^\omega = K' \cdot (\Sigma_I \times \Sigma_O)^\omega$. Furthermore, as $\mathrm{pr}_0(L) = \mathrm{pr}_0(K') \cdot \Sigma_I^\omega$ is universal (Proposition 1), we conclude $\mathrm{pr}_0(K') = \Sigma_I^k$. Here we use that K' only contains words of length k. Thus, for every $\alpha(0) \cdots \alpha(k-1) \in \Sigma_I^k$ there exists $\beta(0) \cdots \beta(k-1) \in \Sigma_O^k$ such that $\binom{\alpha(0)}{\beta(0)} \cdots \binom{\alpha(k-1)}{\beta(k-1)} \in K'$.

This allows Player O to win $\Gamma_f(L)$ for the constant delay function f with $f(0) = k$: every first move of Player I can be answered in the next k rounds such that the resulting word is in K'. Hence, no matter how the play continues, it will be in $L = K' \cdot (\Sigma_I \times \Sigma_O)^\omega$, i.e., Player O wins.

As a corollary, we obtain an upper bound on the complexity of solving delay games whose winning conditions are both reachability and safety. We have just shown that Player O wins $\Gamma_f(L)$ for some f if and only if[5] $K' = \Sigma_I^k$, i.e., we have to check whether for all inputs of length $|\mathcal{A}_R| \cdot |\mathcal{A}_S|$ there is a run of $\mathrm{pr}_0(\mathcal{A}_R)$ that visits an accepting state at least once. This can be done by a polynomial-time alternating Turing machine with a single alternation starting in a universal state, i.e., the problem is in the second level of the polynomial hierarchy.

Corollary 2. *The following problem is in Π_2^{P}: Given a deterministic automaton $\mathcal{A}_R$ and a non-deterministic automaton $\mathcal{A}_S$ with $L_\forall(\mathcal{A}_R) = L_\exists(\mathcal{A}_S)$, does Player O win $\Gamma_f(L_\forall(\mathcal{A}))$ for some f?*

Note that the algorithm does not even process the input $\mathcal{A}_S$, but it needs it (or an equally-sized dummy input) to realize the desired running time.

[5] Actually, we have only shown one direction, but the other one follows directly from Theorem 3.