DELAY GAMES WITH
WMSO+U WINNING CONDITIONS [*]

MARTIN ZIMMERMANN[1]

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. We consider delay games with winning conditions expressed in weak monadic second order logic with the unbounding quantifier, which is able to express (un)boundedness properties.

We show that it is decidable whether the delaying player has a winning strategy using bounded lookahead and give a doubly-exponential upper bound on the necessary lookahead. In contrast, we show that bounded lookahead is not always sufficient: We present a game that can be won with unbounded lookahead, but not with bounded lookahead. Then, we consider such games with unbounded lookahead and show that the exact evolution of the lookahead is irrelevant: The winner is always the same, as long as the initial lookahead is large enough and the lookahead is unbounded.

1991 Mathematics Subject Classification. 68Q45.

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, i.e., 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 [10] 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.

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

1.1. WMSO+U

Recall that the ω-regular languages are exactly those that are definable in (weak[1]) monadic second order logic (MSO and WMSO, respectively) [9]. Recently, Bojańczyk has started a program [1–8, 22] investigating the logic MSO+U, MSO extended with the unbounding quantifier U. A formula $UX\varphi(X)$ is satisfied, if there are arbitrarily large *finite* sets X such that $\varphi(X)$ holds. MSO+U is able to express all ω-regular languages as well as non-regular ones such as $L = \{a^{n_0}ba^{n_1}ba^{n_2}b\cdots \mid \limsup_i n_i = \infty\}$. Decidability of MSO+U remained an open problem until recently: Satisfiability of MSO+U on infinite words is undecidable [6].

Even before this undecidability result was shown, much attention was being paid to fragments of the logic obtained by restricting the power of the second-order quantifiers. In particular, considering weak MSO with the unbounding quantifier turned out to be promising: WMSO+U on infinite words [2] and on infinite trees [8] and WMSO+U with the path quantifier (WMSO+UP) on infinite trees [3] have equivalent automata models with decidable emptiness. Hence, these logics are decidable.

For WMSO+U on infinite words, these automata are called max-automata, deterministic automata with counters whose acceptance conditions are a boolean combination of conditions "counter c is bounded during the run". While processing the input, a counter may be incremented, reset to zero, or the maximum of two counters may be assigned to it (hence the name max-automata). In this work, we investigate delay games with winning conditions given by max-automata, so-called max-regular conditions.

1.2. Delay Games

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

---

[1]In the weak variant of monadic second-order logic, second-order quantification is restricted to finite sets.

every move of her opponent. Delay arises naturally when transmission of data in networks or components with buffers are modeled.

From a more theoretical point of view, uniformization of relations by continuous functions [30–32] can be expressed and analyzed using delay games. We consider games in which two players pick letters from alphabets Σ_I and Σ_O, respectively, thereby producing $\alpha \in \Sigma_I^\omega$ and $\beta \in \Sigma_O^\omega$. Thus, a strategy for the second player induces a mapping $\tau \colon \Sigma_I^\omega \to \Sigma_O^\omega$ turning input sequences $\alpha \in \Sigma_I^\omega$ into output sequences $\beta \in \Sigma_O^\omega$. 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. Then, we say that τ uniformizes L.

In the classical setting of infinite games, in which the players pick letters in alternation, the n-th letter of $\tau(\alpha)$ depends only on the first n letters of α, i.e., τ satisfies a very strong notion of continuity. 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 arbitrary lookahead induces a continuous function (or equivalently, a uniformly continuous function, as Σ^ω is compact). We refer to [20] for a more detailed discussion.

Hosch and Landweber proved that it is decidable whether a game with ω-regular winning condition can be won with bounded lookahead [21]. This result was improved by Holtmann, Kaiser, and Thomas who showed that if a player wins a game with arbitrary lookahead, then she already wins with doubly-exponential bounded lookahead, and gave a streamlined decidability proof yielding an algorithm with doubly-exponential running time [20]. Again, these results were improved by giving a tight exponential upper bound on the necessary lookahead and showing ExpTime-completeness of the solution problem [24]. Going beyond ω-regular winning conditions by considering context-free conditions leads to undecidability and non-elementary lower bounds on the necessary lookahead, even for very weak fragments [18].

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.

In another line of work, Carayol and Löding investigated the case of finite words [13], and Löding and Winter [28] considered the case of finite trees, which are both decidable. However, the nonexistence of MSO-definable choice functions on the infinite binary tree [12, 19] implies that uniformization fails for such trees.

Another application of delay games concerns the existence of Wadge reductions, e.g., reducibility between max-regular languages [11] can be expressed as a max-regular delay game.

1.3. Our Contribution

We start our investigation of max-regular delay games by proving the analogue of the Hosch-Landweber Theorem for max-regular winning conditions: It is decidable whether the delaying player has a winning strategy with bounded lookahead.

Furthermore, we obtain a doubly-exponential upper bound on the necessary looka-head, if this is the case.

WMSO+U is able to express several quantitative winning conditions studied in the literature, e.g., winning conditions in parameterized temporal logics like Prompt-LTL [27], Parametric LTL [33], or Parametric LDL [16], finitary parity and Streett conditions [14], and parity and Streett conditions with costs [17]. Thus, for all these conditions we can decide whether Player O wins a delay game with bounded lookahead.

Our proof consists of a reduction to a delay-free game with a max-regular win-ning condition, i.e., we remove delay. Such games can be solved by expressing them as a satisfiability problem for WMSO+UP on infinite trees: The strategy of one player is an additional labeling of the tree and a path quantifier is able to range over all strategies of the opponent[2]. Satisfiability for WMSO+UP is decidable [3], but the exact complexity of the problem is open.

The reduction itself is an extension of the one used in the ExpTime-algorithm for delay games with ω-regular winning conditions [24] and is based on an equiva-lence relation that captures the behavior of the automaton recognizing the winning condition. However, unlike the relation used for ω-regular conditions, ours is only correct if applied to words of bounded lengths. Thus, we can deal with bounded lookahead, but not with arbitrary lookahead.

We complement the analogue of the Hosch-Landweber Theorem by disproving the analogue of the Holtmann-Kaiser-Thomas Theorem: We give a max-regular delay game that is won by Player O, but only with unbounded lookahead. Thus, not too surprisingly, unbounded lookahead is more powerful than bounded looka-head when it comes to unboundedness conditions.

Finally, we prove that Player O's ability to win a max-regular delay game does not depend on the growth rate of the lookahead, but only on the fact that it grows without bound and on a sufficiently large initial lookahead. This is, to the best of our knowledge, the first such result and should be contrasted with the case of ω-context-free winning conditions, for which a non-elementary growth rate might be necessary for Player O to win [18].

As the analogue of the Holtmann-Kaiser-Thomas Theorem fails, determining the winner of max-regular delay games with respect to arbitrary lookahead does not coincide with determining the winner with respect to bounded lookahead. Hence, we investigate the former problem: We give lower bounds on the complexity and discuss some obstacles one encounters when trying the extend the decidability proof for the bounded case and the undecidability proof for MSO+U satisfiability.

The present paper is a revised and extended version of [34] and is structured as follows: In Section 2, we introduce max-automata and delay games. Then, in Section 3, we introduce the equivalence relations that capture the behavior of max-automata and prove this to be the case. Our results are then presented in the next three sections: The analogue of the Hosch-Landweber Theorem in Section 4, the counterexample to the analogue of the Holtmann-Kaiser-Thomas

---

[2]See Example 1 in [3] for more details.

Theorem in Section 5, and the independence result for unbounded lookahead in Section 6. To conclude, we discuss decidability of delay games with respect to arbitrary lookahead in Section 7 and mention other open problems in Section 8.

2. Definitions

The set of non-negative integers is denoted by $\mathbb{N}$. An alphabet Σ is a non-empty finite set of letters, and Σ^* (Σ^n, Σ^ω) denotes the set of finite words (words of length n, infinite words) over Σ. The empty word is denoted by ε, 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. Given two infinite words $\alpha \in \Sigma_I^\omega$ and $\beta \in \Sigma_O^\omega$ we write $\binom{\alpha}{\beta}$ for the word $\binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)} \cdots \in (\Sigma_I \times \Sigma_O)^\omega$. Analogously, we write $\binom{x}{y}$ for finite words x and y, provided they are of equal length. Finally, the index of an equivalence relation $\equiv$, i.e., the number of its equivalence classes, is denoted by $\mathrm{idx}(\equiv)$.

2.1. Max-Automata

Given a finite set C of counters storing non-negative integers,

$$\mathrm{Ops}(C) = \{c := c + 1, c := 0, c := \max(c_0, c_1) \mid c, c_0, c_1 \in C\}$$

is the set of counter operations over C. A counter valuation over C is a mapping $\nu \colon C \to \mathbb{N}$. By $\nu\pi$ we denote the counter valuation that is obtained by applying a finite sequence $\pi \in \mathrm{Ops}(C)^*$ of counter operations to ν, which is defined as implied by the operations' names.

A max-automaton $\mathcal{A} = (Q, C, \Sigma, q_I, \delta, \ell, \varphi)$ consists of a finite set Q of states, a finite set C of counters, an input alphabet Σ, an initial state q_I, a (deterministic and complete) transition function $\delta \colon Q \times \Sigma \to Q$, a transition labeling[3] $\ell \colon \delta \to \mathrm{Ops}(C)^*$ which labels each transition by a (possibly empty) sequence of counter operations, and an acceptance condition φ, which is a boolean formula over C.

A run of $\mathcal{A}$ on $\alpha \in \Sigma^\omega$ is an infinite sequence

$$\rho = (q_0, \alpha(0), q_1)\,(q_1, \alpha(1), q_2)\,(q_2, \alpha(2), q_3) \cdots \in \delta^\omega \tag{1}$$

with $q_0 = q_I$. Partial (finite) runs on finite words are defined analogously, i.e., $(q_0, \alpha(0), q_1) \cdots (q_{n-1}, \alpha(n-1), q_n)$ is the run of $\mathcal{A}$ on $\alpha(0) \cdots \alpha(n-1)$ starting in q_0. We say that this run ends in q_n. As δ is deterministic, $\mathcal{A}$ has a unique run on every finite or infinite word.

Let ρ be as in (1) and define $\pi_n = \ell(q_n, \alpha(n), q_{n+1})$, i.e., π_n is the label of the n-th transition of ρ. Given an initial counter valuation ν and a counter $c \in C$, we define the sequence

$$\rho_c = \nu(c)\,,\ \nu\pi_0(c)\,,\ \nu\pi_0\pi_1(c)\,,\ \nu\pi_0\pi_1\pi_2(c)\,,\ldots$$

---

[3]Here, and later whenever convenient, we treat δ as relation $\delta \subseteq Q \times \Sigma \times Q$.

of counter values of c reached on the run after applying *all* operations of a transition label. The run ρ of $\mathcal{A}$ on α is accepting, if the acceptance condition φ is satisfied by the variable valuation that maps a counter c to true if, and only if, $\limsup \rho_c$ is finite. Thus, φ can intuitively be understood as a boolean combination of conditions "$\limsup \rho_c < \infty$". Note that the limit superior of ρ_c is independent of the initial valuation used to define ρ_c, which is the reason it is not part of the description of $\mathcal{A}$. We denote the language recognized by $\mathcal{A}$ by $L(\mathcal{A})$. A language is max-regular if it is recognized by some max-automaton.

A parity condition (say min-parity) can be expressed in this framework using a counter for each color that is incremented every time this color occurs and employing the acceptance condition to check that the smallest color whose associated counter is unbounded, is even. Hence, the class of ω-regular languages is contained in the class of max-regular languages.

2.2. Games with Delay

In a delay game, one player gains an advantage over the other by having a lookahead on the opponent's moves. There are at least two equivalent ways of formalizing this interaction. Either, the player given the lookahead may dynamically skip moves while the other may not. In this setting, the rules of the game have to enforce that not almost all moves are skipped to obtain an infinite sequence of (non-skip) moves. In this setting, the evolution of the lookahead is controlled by the strategy of the delaying player and may depend on the history of a play. In the second variant, the evolution of the lookahead is part of the rules of the game and independent of the history of a play. Formally, it is given by a function $f \colon \mathbb{N} \to \mathbb{N} \setminus \{0\}$: One player has to make $f(i)$ moves in round i while the other one only makes one move. If $f(i) > 1$, then the lookahead increases in round i.

The equivalence between these approaches was proven by Holtmann et al. [20] and underlies the Borel determinacy result for delay games [25]. We prefer the second approach, as it allows to specify the degree of lookahead necessary to win a game in a natural way: One can formalize constant, bounded, and unbounded lookahead by restricting the delay functions under consideration.

Formally, a delay function is a mapping $f \colon \mathbb{N} \to \mathbb{N} \setminus \{0\}$, which is said to be bounded, if $f(i) = 1$ for almost all i. A special case of the bounded delay functions are the constant ones, those that satisfy $f(i) = 1$ for all $i > 0$. Note that a constant delay function f is *not* a constant function in the classical sense, but the lookahead granted by f is constant, i.e., the quantity $\sum_{i=0}^{n}(f(i) - 1)$ is constant. The same comment applies to bounded delay functions.

Given a delay function f and an ω-language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$, the game $\Gamma_f(L)$ is played by two players (the male input player "Player I" and the female 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, and only if, the outcome $\binom{\alpha}{\beta}$ is in L, otherwise Player I wins.

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

Theorem 2.1. *Delay games with max-regular winning conditions are determined.*

Proof. This result can either be proven by modelling a delay game with a max-regular winning condition as a delay-free game in a countable arena with a parity winning condition. Such games are determined. Alternatively, one can apply a recent Borel determinacy result for delay games [25], as max-regular languages are Borel [2]. □

Given a max-automaton $\mathcal{A}$, we want to determine whether Player O has a winning strategy for $\Gamma_f(L(\mathcal{A}))$ for some f, and, if yes, what kind of f is sufficient to win, i.e., does constant or bounded lookahead suffice and how large does the constant lookahead have to be.

First, let us remark that bounded and constant lookahead are equivalent, as long as one is only interested in the existence of a delay function that allows Player O to win.

Lemma 2.2. *Let $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$. The following are equivalent:*
  (1) *Player O wins $\Gamma_f(L)$ for some constant f.*
  (2) *Player O wins $\Gamma_f(L)$ for some bounded f.*

Proof. Every constant delay function is bounded, which proves one implication. Now, assume Player O wins $\Gamma_f(L)$ for some bounded delay function f. Let f' be the constant delay function with $f'(0) = f(0) + \sum_{i>0 \colon f(i)>1}(f(i) - 1)$. In every round of $\Gamma_{f'}(L)$, Player O has at least as much lookahead as in the same round of $\Gamma_f(L)$. Thus, she can simulate her winning strategy for $\Gamma_f(L)$ in $\Gamma_{f'}(L)$ and thereby wins this game, too. □

In spite of this equivalence, we consider both classes of delay functions, as it is often simpler to work with constant delay functions (as the lookahead is built up in the first round and then constant) while bounded delay functions yield a more general result. Furthermore, we consider unbounded lookahead in Section 5 and Section 6, the counterpart of bounded lookahead.

3. Equivalence Relations Capturing Max-Automata

Fix a max-automaton $\mathcal{A} = (Q, C, \Sigma, q_I, \delta, \ell, \varphi)$. We generalize notions introduced in [2] to define equivalences over sequences of counter operations and over words over Σ to capture the behavior of $\mathcal{A}$ up to a given precision. The behavior we consider here are the state transformations induced by input words and their effect

on the counter values, e.g., which counters are incremented and which counter values are transferred using the max-operation. In general, there are infinitely many behaviors, e.g., a self-loop with an increment operation allows arbitrarily long increment sequences. To obtain finite equivalence relations, we only keep track of increments up to a given threshold m, which is sufficient to prove our results.

First, we recall and extend some definitions introduced by Bojańczyk to capture the behavior of max-automata. In particular, the notions of a "transfer" and of a "c-trace" are from [2] while we extend his notion of a "transfer with an increment" to an arbitrary, but fixed, number of increments.

First, we recall what it means for a sequence $\pi \in \mathrm{Ops}(C)^*$ to transfer a counter c to a counter d. The empty sequence and the operation $c := c + 1$ transfer every counter to itself. The operation $c := 0$ transfers every counter $c' \neq c$ to itself and the operation $c := \max(c_0, c_1)$ transfers every counter $c' \neq c$ to itself and transfers c_0 and c_1 to c. Finally, if π_0 transfers c to e and π_1 transfers e to d, then $\pi_0 \pi_1$ transfers c to d. If π transfers c to d, then we have $\nu\pi(d) \geq \nu(c)$ for every counter valuation ν, i.e., the value of d after executing π is larger or equal to the value of c before executing π, independently of the initial counter values.

Furthermore, a sequence of counter operations π transfers c to d with $m \geq 0$ increments, if there are counters $e_1, \ldots, e_m$ and a decomposition

$$\pi = \pi_0 \left(e_1 := e_1 + 1\right) \pi_1 \left(e_2 := e_2 + 1\right) \pi_2 \cdots \pi_{m-1} \left(e_m := e_m + 1\right) \pi_m$$

of π such that π_0 transfers c to e_1, π_j transfers e_j to e_{j+1} for every j in the range $1 \leq j < m$, and π_m transfers e_m to d. If π transfers c to d with m increments, then we have $\nu\pi(d) \geq \nu(c) + m$ for every counter valuation ν. Also note that if π transfers c to d with $m > 0$ increments, then it also transfers c to d with m' increments for every $m' \leq m$. Finally, we say that π is a c-trace of length m, if there is a counter c' such that π transfers c' to c with m increments. Thus, if π is a c-trace of length m, then $\nu\pi(c) \geq m$ for every valuation ν.

To illustrate these definitions, consider the following sequence:

$$c := c + 1; \ d := \max(c, d); \ d := d + 1; \ c := 0; \ c := c + 1; \ e := \max(c, d); \ e := e + 1$$

It transfers c to e with three increments; hence, it is an e-trace of length three. Furthermore, it transfers c to d with two increments and is a c-trace of length one, as it has a suffix that transfers c to c with one increment.

As only counter values reached after executing all counter operations of a transition label are considered in the semantics of max-automata, we treat $\Lambda = \{\ell(q, a, q') \mid (q, a, q') \in \delta\}$ as an alphabet. Every word $\lambda \in \Lambda^*$ can be flattened to a word in $\mathrm{Ops}(C)^*$, which is denoted by $\mathrm{flat}(\lambda)$. However, infixes, prefixes, or suffixes of λ are defined with respect to the alphabet Λ. We define $\ell(q, w) \in \Lambda^*$ to be the sequence of elements in Λ labeling the run $\rho(q, w)$.

Let $\lambda \in \Lambda^*$ be a sequence of transition labels and let $\pi \in \mathrm{Ops}(C)^*$ be a sequence of counter instructions. We say that λ ends with π, if π is a suffix of $\mathrm{flat}(\lambda)$ and that λ contains π, if λ has a prefix that ends with π. Containment of π in an infinite sequence of transition labels is defined similarly.

Next, we need to introduce some notation to deal with runs of $\mathcal{A}$. Given a state q and $w \in \Sigma^* \cup \Sigma^\omega$, let $\rho(q, w)$ be the run of $\mathcal{A}$ on w starting in q. If w is finite, then $\delta^*(q, w)$ denotes the state $\rho(q, w)$ ends with. The transition profile of $w \in \Sigma^*$ is the mapping $q \mapsto \delta^*(q, w)$.

Now, we can lift the notions of "ending with π" and "containing π" to runs: Let ρ be a finite run of $\mathcal{A}$ and let $\pi \in \mathrm{Ops}(C)^*$. We say that ρ ends with π, if $\ell(\rho)$ ends with π. Similarly, a finite or infinite run ρ contains π, if $\ell(\rho)$ contains π. These definitions account for the fact that only counter values reached after executing all counter operations of a transition label are considered in the semantics of max-automata.

Lemma 3.1 ([2])**.** *Let ρ be a run of $\mathcal{A}$ and c a counter. Then, $\limsup \rho_c = \infty$ if, and only if, ρ contains arbitrarily long c-traces.*

We use the notions of transfer (with increments) to define the equivalence relations that capture $\mathcal{A}$'s behavior, i.e., the c-traces that occur during a run. Let $\pi = \pi_0 \pi_1 \pi_2 \cdots$ be an infinite sequence of blocks $\pi_i \in \mathrm{Ops}(C)^*$ of counter operations. A c-trace contained in π can either be contained in some block π_i or span multiple π_i. In the latter case, it starts with a suffix of some π_i, continues throughout several segments as transfers, and then ends with a prefix of some $\pi_{i'}$.

The following definition captures all these cases. Fix some $m \geq 0$. We say that $\lambda, \lambda' \in \Lambda^*$ are m-equivalent[4], denoted by $\lambda \equiv_{\mathrm{ops}}^m \lambda'$, if for all counters c and d and for all m' in the range $0 \leq m' \leq m$:

(1) λ contains a c-trace of length m' if, and only if, λ' contains a c-trace of length m',
(2) λ ends with a c-trace of length m' if, and only if, λ' ends with a c-trace of length m',
(3) the flattening of λ transfers c to d with m' increments if, and only if, the flattening of λ' transfers c to d with m' increments, and
(4) λ has a prefix whose flattening transfers c to d with m' increments if, and only if, λ' has a prefix whose flattening transfers c to d with m' increments.

Using this, we define two words $x, x' \in \Sigma^*$ to be m-equivalent, denoted by $x \equiv_{\mathcal{A}}^m x'$, if they have the same transition profile and if $\ell(q, x) \equiv_{\mathrm{ops}}^m \ell(q, x')$ for all states q.

Recall that a congruence is an equivalence relation $\equiv$ over Σ^* such that $x \equiv y$ implies $xz \equiv yz$ for every $z \in \Sigma^*$.

Lemma 3.2. *Let $\mathcal{A}$ be a max-automaton with n states and k counters and let $m \in \mathbb{N}$.*

---

[4]The definition of the equivalence relation used in the conference version of this paper [34] differs in several aspects from the one we present here, as we consider more general problems here. In particular, we added the first requirement, which is important when analyzing games with unbounded lookahead, which we do here, but did not do in the conference version. Secondly, we added the fourth requirement to fix a bug in the proof of Item (1) of the analogue of Lemma 3.3 in the conference version. Finally, we parameterized the equivalence relation with $m \geq 0$ while we only considered the case $m = 1$ in the conference version. Again, this is necessary to reason about unbounded lookahead. Note that the definition presented here slightly increases the upper bounds on the index presented in Lemma 3.2 in comparison to the old one.

(1) $\lambda \equiv_{\text{ops}}^{m} \lambda'$ implies $\lambda \equiv_{\text{ops}}^{m'} \lambda'$ for every $m' \leq m$.

(2) $x \equiv_{\mathcal{A}}^{m} x'$ implies $x \equiv_{\mathcal{A}}^{m'} x'$ for every $m' \leq m$.

(3) $\equiv_{\text{ops}}^{m}$ is a congruence.

(4) $\equiv_{\mathcal{A}}^{m}$ is a congruence.

(5) The index of $\equiv_{\text{ops}}^{m}$ is at most $2^{2(k^2+k)\log(m+2)}$.

(6) The index of $\equiv_{\mathcal{A}}^{m}$ is at most $2^{n(\log(n)+2(k^2+k)\log(m+2))}$.

Proof. The first two items follow trivially from the definition of $\equiv_{\text{ops}}^{m}$. Thus, we only consider the latter four items.

(3) Let $\lambda \equiv_{\text{ops}}^{m} \lambda'$ and let $\pi \in \Lambda$ (note that we treat π as a letter from Λ, although it is also a sequence of counter operations). We show $\lambda\pi \equiv_{\text{ops}}^{m} \lambda'\pi$. Then, an inductive application proves that $\equiv_{\text{ops}}^{m}$ is a congruence.

First, assume $\lambda\pi$ contains a c-trace of length for some $m' \leq m$, i.e., $\lambda\pi$ has an infix λ_0 whose flattening has a suffix π_0 that is a c-trace of length m'. If λ_0 is an infix of λ, then $\lambda \equiv_{\text{ops}}^{m} \lambda'$ implies that λ' has an infix with the same property. The other trivial case is when λ_0 is equal to π. Thus, it remains to consider the case where λ_0 is a suffix of $\lambda\pi$ of length at least two (recall that we treat π as one letter, i.e., λ_0 contains at least one letter from λ). Thus, π_0 can be decomposed into two parts, one that is a c'-trace of length m_0 and is a suffix of the flattening of λ, and another one that is equal to π (treated as a sequence of counter operations now), which transfers c' to c with m_1 increments. Furthermore, we have $m_0 + m_1 = m' \leq m$.

Due to $\lambda \equiv_{\text{ops}}^{m} \lambda'$, we conclude that the flattening of λ' has a suffix that is a c'-trace of length m_0. Combining this suffix with π, we obtain a suffix of the flattening of $\lambda'\pi$ that is a c-trace of length m'. Hence, there is also an infix of $\lambda'\pi$ whose flattening has a suffix that is a c-trace of length m', i.e., $\lambda'\pi$ contains a c-trace of length m'.

The argument where $\lambda'\pi$ has such an infix is symmetric and the reasoning for the other three properties in the definition of $\equiv_{\text{ops}}^{m}$ is analogous.

(4) Due to $\delta^*(q, xz) = \delta(\delta^*(q, x), z)$, having the same transition profile is a congruence. This, and $\equiv_{\text{ops}}^{m}$ being a congruence imply that $\equiv_{\mathcal{A}}^{m}$ is a congruence, too.

(5) An $\equiv_{\text{ops}}^{m}$ equivalence class is uniquely characterized by the following properties:

- For every counter c, by the largest $m' \leq m$ such that its elements contain a c-trace of length m' (note that every word contains a c-trace of length 0, e.g., the empty sequence of operations).
- For every counter c, by the largest $m' \leq m$ such that its elements end with a c-trace of length m' (note that every word ends with a c-trace of length 0, e.g., the empty sequence of operations).
- For every pair (c, d) of counters, whether the flattenings of its elements transfer c to d, and if yes by the largest $m' \leq m$ such that the transfer has m' increments.

- For every pair (c, d) of counters, whether its elements have a prefix whose flattening transfers c to d, and if yes by the largest $m' \leq m$ such that the transfer has m' increments.

Thus, an equivalence class is induced by two mappings from C to $\{0, 1, \ldots, m\}$ and two mappings from $C \times C$ to $\{\perp, 0, 1, \ldots, m\}$, where $\perp$ encodes that no such transfer exists. The number of quadruples of such mappings is bounded by

$$(m + 2)^{2(k^2+k)} = 2^{2(k^2+k)\log(m+2)}.$$

(6) An equivalence class of $\equiv_{\mathcal{A}}^m$ is uniquely characterized by a transition profile and, for every state q, by the $\equiv_{\text{ops}}^m$ equivalence class of the sequence of counter operations encountered along the run starting in q. Thus, the class is characterized by a mapping from Q to pairs of a state and an $\equiv_{\text{ops}}^m$ class. Thus, the index of $\equiv_{\mathcal{A}}^m$ is bounded by the number of such mappings, i.e., by

$$(n \cdot \text{idx}(\equiv_{\text{ops}}^m))^n = 2^{\log\left((n2^{2(k^2+k)\log(m+2)})^n\right)} = 2^{n\left(\log(n)+2(k^2+k)\log(m+2)\right)}. \qquad \square$$

Next, we show that we can take any infinite word $x_0 x_1 x_2 \cdots$ with $x_i \in \Sigma^*$ and replace each x_i by an *equivalent* x_i' without changing membership in $L(\mathcal{A})$. We present two variants of this replacement, which differ in the formalization of the equivalence. One variant is for the case of bounded lookahead and one for unbounded lookahead. In the former case, we have to require that the lengths of the x_i and the lengths of the x_i' are bounded. Then, one can replace each x_i by an $\equiv_{\mathcal{A}}^1$-equivalent x_i', as the error introduced by the replacement is bounded due to the bound on the word lengths. For the unbounded case, we have to capture the evolution of the counters properly with the imprecise equivalence relations $\equiv_{\mathcal{A}}^m$. Here, we require that the x_i and the x_i' are $\equiv_{\mathcal{A}}^m$-equivalent for m tending to infinity. Formally, a sequence $(r_i)_{i \in \mathbb{N}}$ of natural numbers is a (convergence) rate, if it is weakly increasing and unbounded, i.e., $r_i \leq r_{i+1}$ for every i and $\sup_i r_i = \infty$. Then, we are able to replace each x_i by some $\equiv_{\mathcal{A}}^{r_i}$-equivalent x_i'.

Lemma 3.3. *Let $(x_i)_{i \in \mathbb{N}}$ and $(x_i')_{i \in \mathbb{N}}$ be two sequences of words over Σ^*. Define $x = x_0 x_1 x_2 \cdots$ and $x' = x_0' x_1' x_2' \cdots$.*

  (1) If $\sup_i |x_i| < \infty$, $\sup_i |x_i'| < \infty$, and $x_i \equiv_{\mathcal{A}}^1 x_i'$ for all i, then $x \in L(\mathcal{A})$ if, and only if, $x' \in L(\mathcal{A})$.
  (2) If there is a rate $(r_i)_{i \in \mathbb{N}}$ such that $x_i \equiv_{\mathcal{A}}^{r_i} x_i'$ for all i, then $x \in L(\mathcal{A})$ if, and only if, $x' \in L(\mathcal{A})$.

Proof. We start by introducing some notation that is used in both items of the proof, based on the fact that we have $x_i \equiv_{\mathcal{A}}^0 x_i'$ for all i in both cases, due to Item (2) of Lemma 3.2.

Let $\rho = \rho(q_I, x)$ be the run of $\mathcal{A}$ on x and let $\rho' = \rho(q_I, x)$ be the run of $\mathcal{A}$ on x'. Furthermore, let $q_i = \delta^*(q_I, x_0 \cdots x_{i-1})$ and $q_i' = \delta^*(q_I, x_0' \cdots x_{i-1}')$ be the states reached after processing the prefixes $x_0 \cdots x_{i-1}$ and $x_0' \cdots x_{i-1}'$, respectively. By definition of $\equiv_{\mathcal{A}}^0$, we obtain $q_i = q_i'$ for every i.

Furthermore, let $\lambda_i = \ell(q_i, x_i)$ be the sequence of counter operations labeling the run of $\mathcal{A}$ on x_i starting in q_i, which ends in q_{i+1}. The sequences $\lambda_i' = \ell(q_i', x_i')$ labeling the runs on the x_i' are defined analogously.

In both cases, we show that ρ contains arbitrarily long c-traces if, and only if, ρ' contains arbitrarily long c-traces. Due to Lemma 3.1, this suffices to show that ρ is accepting if, and only if, ρ' is accepting. Furthermore, due to symmetry, it suffices to show one direction of the equivalence. Thus, assume ρ contains arbitrarily long c-traces and pick $m' \in \mathbb{N}$ arbitrarily. We show the existence of a c-trace of length m' contained in ρ'. From now on, we have to consider both items separately.

(1) Here, we take a c-trace in ρ of length $m > m'$ for some sufficiently large m and show that the corresponding part of ρ' contains a c-trace of length m'. To begin, we note that $x_i \equiv_{\mathcal{A}}^1 x_i'$ and $q_i = q_i'$ implies that λ_i and λ_i' are $\equiv_{\mathrm{ops}}^1$-equivalent as well. Furthermore, define $b = \sup_i |x_i|$, which is well-defined due to our assumption, and define $m = (m' + 1) \cdot o \cdot b$, where $o = \max_{\pi \in \Lambda} |\pi|$ is the maximal length of a sequence of operations labeling a transition (viewed as a word over $\mathrm{Ops}(C)$). Each λ_i can contribute at most $|\mathrm{flat}(\lambda_i)|$ increments to a c-trace that subsumes λ_i, which is bounded by $|\mathrm{flat}(\lambda_i)| \leq o \cdot b$.

Now, we pick i such that $\lambda_0 \cdots \lambda_i$ contains a c-trace of length m. By the choice of m, this trace spans several λ_i, i.e., there are counters $c_{s+1}, c_{s+2}, \ldots, c_i$ such that

- the flattening of λ_s has a suffix that is a c_{s+1}-trace,
- the flattening of λ_j transfers c_j to c_{j+1} for every j in the range $s < j < i$, and
- λ_i has a prefix whose flattening transfers c_i to c.

By the choice of m we know that we can pick the counters $c_{s+1}, c_{s+2}, \ldots, c_i$ so that at least m' of the transfers between them are actually transfers with at least one increment, as every transfer contains at most $b \cdot o$ increments.

The equivalence of λ_j and λ_j' implies that λ_j' realizes the same transfers (with at least one increment) as λ_j. Hence, $\lambda_0' \cdots \lambda_i'$ contains a c-trace of length m'.

(2) Here, we can take a c-trace of length m' in ρ appearing after a sufficiently large prefix of ρ so that the rate r_i is large enough to imply that the $\equiv_{\mathrm{ops}}^{r_i}$-equivalent part in ρ' contains a c-trace of length m' as well. Again, we begin by noting that we have $\lambda_i \equiv_{\mathrm{ops}}^{r_i} \lambda_i'$ for every i, due to $x_i \equiv_{\mathcal{A}}^{r_i} x_i'$ and $q_i = q_i'$.

Now, fix a c-trace of length m' in ρ. As there are arbitrarily long such traces, we can pick one that is contained in some infix $\lambda_s \cdots \lambda_i$ with $m' \leq r_s \leq r_i$.

If $s = i$, then the complete c-trace is contained in λ_s. Thus, the first requirement in the definition of $\equiv_{\mathrm{ops}}^{r_i}$ yields that λ_s' contains a c-trace of the same length.

If $s < i$, then the reasoning is similar to Item (1) above: There are counters $c_{s+1}, c_{s+2}, \ldots, c_i$ and natural numbers $m_s, m_{s+1}, \ldots, m_i$ with $m_s + m_{s+1} + \cdots + m_i = m'$ such that

- the flattening of λ_s has a suffix that is a c_{s+1}-trace of length m_s,
- the flattening of λ_j transfers c_j to c_{j+1} with m_j increments for every j in the range $s < j < i$, and
- λ_i has a prefix whose flattening transfers c_i to c with m_i increments.

The equivalence of λ_j and λ'_j implies that λ'_j realizes the same transfers with the same number of increments as λ_j. Hence, ρ' contains a c-trace of length m'. $\qquad\square$

Note that the first item of the lemma does not hold if we drop the boundedness requirements on the lengths of the x_i and the x'_i.

The $\equiv_{\mathcal{A}}^m$ classes are regular and *trackable* on-the-fly by a deterministic finite automaton (DFA) $\mathcal{T}_m$ due to $\equiv_{\mathcal{A}}^m$ being a congruence.

Lemma 3.4. *There is a DFA $\mathcal{T}_m$ with set of states $\Sigma^*/\equiv_{\mathcal{A}}^m$ such that the run of $\mathcal{T}_m$ on $w \in \Sigma^*$ ends with state $[w]_{\equiv_{\mathcal{A}}^m}$.*

Proof. Define $\mathcal{T}_m = (\Sigma^*/\equiv_{\mathcal{A}}^m, \Sigma, [\varepsilon]_{\equiv_{\mathcal{A}}^m}, \delta_{\mathcal{T}_m}, \emptyset)$ where $\delta_{\mathcal{T}_m}([x]_{\equiv_{\mathcal{A}}^m}, a) = [xa]_{\equiv_{\mathcal{A}}^m}$, which is independent of the representative x and based on the fact that $\equiv_{\mathcal{A}}^m$ is a congruence. An induction over $|w|$ shows that $\mathcal{T}_m$ has the desired properties. $\qquad\square$

In particular, every $\equiv_{\mathcal{A}}^m$ equivalence class is regular and recognized by the DFA obtained from $\mathcal{T}_m$ by making the class to be recognized the only final state.

For the remainder of this section, we assume $\Sigma = \Sigma_I \times \Sigma_O$. We denote the projection of $\Sigma_I \times \Sigma_O$ to Σ_I by $\pi_I(\cdot)$, an operation we lift to words and languages over $\Sigma_I \times \Sigma_O$ in the usual way. Now, for each equivalence relation $\equiv_{\mathcal{A}}^m$ over $(\Sigma_I \times \Sigma_O)^*$ we define its projection[5] $=_{\mathcal{A}}^m$ over Σ_I^* via $x =_{\mathcal{A}}^m x'$ if, and only if, for all $\equiv_{\mathcal{A}}^m$ classes S: $x \in \pi_I(S)$ if, and only if, $x' \in \pi_I(S)$.

Remark 3.5. $\mathrm{idx}(=_{\mathcal{A}}^m) \le 2^{\mathrm{idx}(\equiv_{\mathcal{A}}^m)}$.

Furthermore, every $=_{\mathcal{A}}^m$ equivalence class is regular: We have

$$[x]_{=_{\mathcal{A}}^m} = \bigcap\nolimits_{S \in (\Sigma_I \times \Sigma_O)^*/\equiv_{\mathcal{A}}^m \,:\, x \in \pi_I(S)} \pi_I(S) \cap \bigcap\nolimits_{S \in (\Sigma_I \times \Sigma_O)^*/\equiv_{\mathcal{A}}^m \,:\, x \notin \pi_I(S)} \Sigma_I^* \setminus \pi_I(S),$$

where each projection $\pi_I(S)$ and each complemented projection $\Sigma_I^* \setminus \pi_I(S)$ is recognized by a DFA of size $2^{\mathrm{idx}(\equiv_{\mathcal{A}}^m)}$. All these DFA's share the same set of states. Thus, $[x]_{=_{\mathcal{A}}^m}$ is recognized by a DFA of size $2^{\mathrm{idx}(\equiv_{\mathcal{A}}^m)}$ as well. In particular, we have the following bound that is applied in the next section, which stems from the fact that a DFA with s states that recognizes a word w of length $|w| \ge s$ recognizes an infinite language.

Remark 3.6. Recall that n denotes the number of states and k the number of counters of $\mathcal{A}$ and let x be in a finite equivalence class of $=_{\mathcal{A}}^0$. Then, we have $|x| < 2^{\mathrm{idx}(\equiv_{\mathcal{A}}^0)} = 2^{2^{n(\log(n)+2(k^2+k))}}$.

4. Max-regular Delay Games with Bounded Lookahead

In this section, we prove the analogue of the Hosch-Landweber Theorem for max-regular winning conditions: Given a max-automaton $\mathcal{A}$, it is decidable whether Player O wins $\Gamma_f(L(\mathcal{A}))$ for some constant f. The proof consists of a reduction

---

[5]The notation $=_{\mathcal{A}}^m$ should not be understood as denoting equality, but merely as having projected away one bar from $\equiv_{\mathcal{A}}^m$.

to a delay-free game with max-regular winning condition. The winner of such a game can be determined effectively by a reduction to the satisfiability problem for WMSO+UP. As the complexity of the satisfiability problem is open, even when already starting with an automaton instead of a formula, we also obtain a decidability result without any upper bound on the complexity. We come back to this issue in Section 8.

The delay-removal is similar to the one in the ω-regular case that forms the foundation for the exponential-time algorithm solving such games [24]. Intuitively, instead of picking words over their alphabet, Player I picks $=_{\mathcal{A}}^1$ equivalence classes and Player O picks compatible $\equiv_{\mathcal{A}}^1$ classes. To account for the lookahead, Player I is aways one move ahead in the delay-free game.

By picking representatives of the $\equiv_{\mathcal{A}}^1$ classes picked by Player O, one obtains a word whose membership in $L(\mathcal{A})$ determines the winner. The error introduced by using the imprecise equivalence relation is bounded, as we only consider constant delay functions. The correctness of this construction is based on Item (1) of Lemma 3.3, which shows that such a bounded error is negligible when it comes to satisfying the acceptance condition of a max-automaton.

To obtain small bounds on the necessary lookahead, we modify $\mathcal{A}$ so that it keeps track of the $\equiv_{\mathcal{A}}^1$ class of the input it processes and then work with the projection of the modified automaton $\mathcal{A}$ instead of projecting the equivalence relation. This approach yielded the exponential improvement between the algorithms presented in [20] and [24] for solving ω-regular delay games.

Theorem 4.1. *The following problem is decidable: Given a max-automaton $\mathcal{A}$, does Player O win $\Gamma_f(L(\mathcal{A}))$ for some constant delay function f?*

Let $\mathcal{A} = (Q, C, \Sigma_I \times \Sigma_O, q_I, \delta, \ell, \varphi)$ be a max-automaton and let the tracking automaton $\mathcal{T}_1 = ((\Sigma_I \times \Sigma_O)/\equiv_{\mathcal{A}}^1, \Sigma_I \times \Sigma_O, [\varepsilon]_{\equiv_{\mathcal{A}}^1}, \delta_{\mathcal{T}_1}, \emptyset)$ be defined as in Lemma 3.4. In this section, for the sake of readability, we denote the $\equiv_{\mathcal{A}}^1$ equivalence class of w by $[w]$ and do not make use of the other equivalence relations $\equiv_{\mathcal{A}}^m$ for $m \neq 1$. Furthermore, we denote equivalence classes using the letter S.

We define the product $\mathcal{P} = (Q_{\mathcal{P}}, C, \Sigma_I \times \Sigma_O, q_I^{\mathcal{P}}, \delta_{\mathcal{P}}, \ell_{\mathcal{P}}, \varphi)$ of $\mathcal{A}$ and $\mathcal{T}_1$, which is a max-automaton, where

- $Q_{\mathcal{P}} = Q \times ((\Sigma_I \times \Sigma_O)/\equiv_{\mathcal{A}}^1)$,
- $q_I^{\mathcal{P}} = (q_I, [\varepsilon])$,
- $\delta_{\mathcal{P}}((q, S), a) = (\delta(q, a), \delta_{\mathcal{T}_1}(S, a))$ for $q \in Q$, a class $S \in (\Sigma_I \times \Sigma_O)/\equiv_{\mathcal{A}}^1$, and a letter $a \in \Sigma_I \times \Sigma_O$, and
- $\ell_{\mathcal{P}}((q, S), a, (q', S')) = \ell(q, a, q')$.

Let

$$n = |Q_{\mathcal{P}}| = |Q| \cdot \mathrm{idx}(\equiv_{\mathcal{A}}^1) \leq |Q| \cdot 2^{|Q|(\log(|Q|) + 4(k^2 + k))}.$$

We have $L(\mathcal{P}) = L(\mathcal{A})$, since acceptance only depends on the component $\mathcal{A}$ of $\mathcal{P}$. However, we are interested in partial runs of $\mathcal{P}$, as the component $\mathcal{T}_1$ keeps track of the equivalence class of the input processed by $\mathcal{P}$.

Remark 4.2. Let $w \in (\Sigma_I \times \Sigma_O)^*$ and let $(q_0, S_0)(q_1, S_1) \cdots (q_{|w|}, S_{|w|})$ be the run of $\mathcal{P}$ on w from some state (q_0, S_0) with $S_0 = [\varepsilon]$. Then, $q_0 q_1 \cdots q_{|w|}$ is the run of $\mathcal{A}$ on w starting in q_0 and $S_{|w|} = [w]$.

In the following, we work with partial functions r from $Q_\mathcal{P}$ to $2^{Q_\mathcal{P}}$, where we denote the domain of r by $\mathrm{dom}(r)$. Intuitively, we use such a function 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 some $i < j$, i.e., the lookahead is $\alpha(i+1) \cdots \alpha(j)$. Then, we can determine the state q that $\mathcal{P}$ reaches when processing $\binom{\alpha(0)}{\beta(0)} \cdots \binom{\alpha(i)}{\beta(i)}$, but the automaton cannot process $\alpha(i+1) \cdots \alpha(j)$, since Player O has not yet provided her moves $\beta(i+1) \cdots \beta(j)$. However, we can determine which states Player O can enforce by picking an appropriate completion. These are contained in $r(q)$.

To formalize this, we first treat $\mathcal{P}$ as a DFA, i.e., we ignore the transition labeling and the acceptance condition. Then, we project away the second component of the alphabet to obtain a non-deterministic automaton over the alphabet Σ_I, denoted by $\pi_I(\mathcal{P})$. Finally, we apply the powerset construction to determinize the automaton. Let $\delta_{\mathrm{pow}} : 2^{Q_\mathcal{P}} \times \Sigma_I \to 2^{Q_\mathcal{P}}$ be the transition function of the powerset automaton, i.e., $\delta_{\mathrm{pow}}(P, a) = \bigcup_{q \in P} \bigcup_{b \in \Sigma_O} \delta_\mathcal{P}\left(q, \binom{a}{b}\right)$. As usual, we extend δ_{pow} to $\delta_{\mathrm{pow}}^* : 2^{Q_\mathcal{P}} \times \Sigma_I^* \to 2^{Q_\mathcal{P}}$ via $\delta_{\mathrm{pow}}^*(P, \varepsilon) = P$ and $\delta_{\mathrm{pow}}^*(P, wa) = \delta_{\mathrm{pow}}(\delta_{\mathrm{pow}}^*(P, w), a)$.

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

$$r_w^D(q, S) = \delta_{\mathrm{pow}}^*(\{(q, [\varepsilon])\}, w) \ ,$$

i.e., we collect all states (q', S') reachable from $(q, [\varepsilon])$ (note that the second component is the equivalence class of the empty word while the class S from the argument is ignored) via a run of the projected automaton $\pi_I(\mathcal{P})$ on w. Thus, if $(q', S') \in r_w^D(q, S)$, then there is a word w' whose projection is w and with $[w'] = S'$ such that the run of $\mathcal{A}$ on w' leads from q to q'. Hence, 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' and such that it is a representative of S'.

We call w a witness for a partial function $r : Q_\mathcal{P} \to 2^{Q_\mathcal{P}}$, 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 4.3. *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 $r, r' \in \mathfrak{R}$ such that $r \neq r'$ and $\mathrm{dom}(r) = \mathrm{dom}(r')$. Then, $W_r \cap W_{r'} = \emptyset$.

(3) Let r be a partial function from $Q_\mathcal{P}$ to $2^{Q_\mathcal{P}}$. Then, W_r is recognized by a DFA with at most 2^{n^2} states.

(4) Let $r \in \mathfrak{R}$. Then, W_r contains a word w with $k \leq |w| \leq k + 2^{n^2}$ for every k.

(5) Let $D \subseteq Q_\mathcal{P}$ 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 $w \in W_r$.

Proof. The first statement follows from completeness of the automaton $\mathcal{P}$ while the second one follows from the definition of r_w^D, which is uniquely determined by w and D. Hence, a fixed w cannot witness two different functions r and r' with the same domain.

To prove the third statement, fix some partial function r from $Q_{\mathcal{P}}$ to $2^{Q_{\mathcal{P}}}$ with domain $D = \{(q_1, S_1), \ldots, (q_{|D|}, S_{|D|})\}$. Then, the product of $|D|$ copies of the powerset automaton of $\pi_I(\mathcal{P})$ (ignoring the transition labeling and the acceptance condition) with the initial state $(\{(q_1, [\varepsilon])\}, \ldots, \{(q_{|D|}, [\varepsilon])\})$ and the unique accepting state $(r(q_1, S_1), \ldots, r(q_{|D|}, S_{|D|}))$ recognizes the witness language W_r. As $|D| \leq n$, the automaton has at most $(2^n)^{|D|} \leq 2^{n^2}$ states.

The fourth statement is follows immediately by a simple pumping argument from the third one: Every finite automaton with s states recognizing an infinite language recognizes a word w of length $k \leq |w| \leq k + s$ for every k.

For proving the last statement, we fix some non-empty D and some w of length at least 2^{n^2}. Define $r = r_w^D$, which implies $w \in W_r$ by definition. As just shown, there exists an automaton recognizing W_r with at most $2^{n^2} \leq |w|$ many states. Thus, the accepting run of the automaton on w contains a state-repetition. Hence, W_r is infinite, i.e., $r \in \mathfrak{R}$. $\square$

Due to Items (2) and (5), we can define for every non-empty $D \subseteq Q_{\mathcal{P}}$ a function r_D that maps words $w \in \Sigma_I^*$ with $|w| \geq 2^{n^2}$ to the unique function r with $\mathrm{dom}(r) = D$ and $w \in W_r$. This is used later in the proof.

Now, we define an abstract game $\mathcal{G}(\mathcal{A})$ between Player I and Player O that is played in rounds $i = 0, 1, 2, \ldots$: In each round, Player I picks a function from $\mathfrak{R}$ and then Player O picks a state q of $\mathcal{P}$. In round 0, Player I has to pick r_0 subject to constraint (C1): $\mathrm{dom}(r_0) = \{q_I^{\mathcal{P}}\}$. Then, Player O has to pick a state $q_0 \in \mathrm{dom}(r_0)$ (which implies $q_0 = q_I^{\mathcal{P}}$). 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 subject to constraint (C2): $\mathrm{dom}(r_i) = r_{i-1}(q_{i-1})$. Then, Player O has to pick a state $q_i \in \mathrm{dom}(r_i)$. Both players can always move: Player I can, as $r_{i-1}(q_{i-1})$ is always non-empty (Item (1) of Lemma 4.3) and thus the domain of some $r \in \mathfrak{R}$ (Item (5) of Lemma 4.3) and Player O can, as the domain of every $r \in \mathfrak{R}$ is non-empty by construction.

The resulting play is the sequence $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$. Let $q_i = (q_i', S_i)$ for every i, i.e., S_i is an $\equiv_{\mathcal{A}}^1$ equivalence class. Let $x_i \in S_i$ for every i such that $\sup_i |x_i| < \infty$, i.e., we pick representatives whose lengths are bounded. Such a sequence can always be found as $\equiv_{\mathcal{A}}^1$ has finite index. Player O wins the play if the word $x_0 x_1 x_2 \cdots$ is accepted by $\mathcal{A}$. Due to Item (1) of Lemma 3.3, this definition is independent of the choice of the representatives x_i. Hence, the winner of the play only depends on the sequence $S_0 S_1 S_2 \cdots$.

A strategy for Player I is a function τ_I' mapping the empty play prefix to a function r_0 subject to constraint (C1) and mapping a non-empty play prefix $r_0 q_0 \cdots r_{i-1} q_{i-1}$ ending in a state to a function r_i subject to constraint (C2). On the other hand, a strategy for Player O maps a play prefix $r_0 q_0 \cdots r_i$ ending in a function 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 $i \geq 0$. Dually, the play is consistent with τ_O', if $q_i = \tau_O'(r_0 q_0 \cdots r_i)$ for every $i \geq 0$. A strategy is winning for Player p, if every play that is consistent with this strategy is winning for her. As usual, we say that Player p wins $\mathcal{G}(\mathcal{A})$, if she has a winning strategy.

In the proof of Theorem 4.1, we construct an explicit variant of $\mathcal{G}(\mathcal{A})$ and show that its winning condition is max-regular. As a corollary, we obtain determinacy of $\mathcal{G}(\mathcal{A})$. But first we prove that $\mathcal{G}(\mathcal{A})$ captures the existence of a bounded delay function f such that Player O wins $\Gamma_f(L(\mathcal{A}))$.

Lemma 4.4. *Player O wins $\Gamma_f(L(\mathcal{A}))$ for some constant delay function f if, and only if, Player O wins $\mathcal{G}(\mathcal{A})$.*

Proof. For the sake of readability, we write Γ instead of $\Gamma_f(L(\mathcal{A}))$, as long as f is clear from context. Similarly, we write $\mathcal{G}'$ instead of $\mathcal{G}'(\mathcal{A})$.

First, assume Player O has a winning strategy τ_O for $\Gamma_f(L(\mathcal{A}))$ for some constant delay function f. We construct a winning strategy τ_O' for Player O in $\mathcal{G}$ via 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 responded to by Player O by picking $q_I^{\mathcal{P}} = \tau_O'(r_0)$, and let r_1 be Player I's response to that move. Let $w_0 \in W_{r_0}$ and $w_1 \in W_{r_1}$ be witnesses for the functions picked by Player I. Due to Item (4) of Lemma 4.3, we can choose w_0 and w_1 with $f(0) \leq |w_0|, |w_1| \leq f(0) + 2^{n^2}$. We simulate the play prefix $r_0 q_0 r_1$ in Γ, where $q_0 = q_I^{\mathcal{P}}$: 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. We obtain $|\beta(0) \cdots \beta(\ell_1 - f(0))| \geq |w_0|$, due to $f(0) \leq |w_1|$.

Thus, we are in the following situation for $i = 1$: In $\mathcal{G}$, we have a play prefix $r_0 q_0 \cdots r_{i-1} q_{i-1} r_i$ and in Γ, Player I has picked $w_0 w_1 \cdots w_i = \alpha(0) \cdots \alpha(\ell_i - 1)$ and Player O has picked $\beta(0) \cdots \beta(\ell_i - f(0))$ according to τ_O, where $|\beta(0) \cdots \beta(\ell_i - f(0))| \geq |w_0 \cdots w_{i-1}|$. Furthermore, w_j is a witness for r_j for every $j \leq i$.

In this situation, let q_i be the state of $\mathcal{P}$ that is reached when processing w_{i-1} and the corresponding moves of Player O, i.e., the word

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

starting in state $(q_{i-1}', [\varepsilon])$, where $q_{i-1} = (q_{i-1}', S_{i-1})$.

By definition of r_{i-1}, we have $q_i \in r_{i-1}(q_{i-1})$, i.e., q_i is a legal move for Player O in $\mathcal{G}$ to extend the play prefix $r_0 q_0 \cdots r_{i-1} q_{i-1} r_i$. Thus, we define $\tau_O'(r_0 q_0 \cdots r_{i-1} q_{i-1} r_i) = q_i$. 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 $f(0) \leq |w_{i+1}| \leq f(0) + 2^{n^2}$. Going back to Γ, let Player I pick $w_{i+1} = \alpha(\ell_i) \cdots \alpha(\ell_{i+1} - 1)$ as his next moves and let $\beta(\ell_i - f(0) + 1) \cdots \beta(\ell_{i+1} - f(0))$ be the response of Player O according to τ_O. Then, we are in the situation as described in the previous paragraph, which concludes the definition of τ_O'.

It remains to show that the strategy τ_O' is winning for Player O in $\mathcal{G}$. Consider a play $r_0q_0r_1q_1r_2q_2\cdots$ that is consistent with τ_O' and let $w = \binom{\alpha}{\beta}$ be the corresponding outcome constructed as in the simulation described above. Let $q_i = (q_i', S_i)$, i.e., q_i' is a state of our original automaton $\mathcal{A}$. A straightforward inductive application of Remark 4.2 shows that q_i' is the state that $\mathcal{A}$ reaches after processing w_i and the corresponding moves of Player O, i.e.,

$$
x_i = \begin{pmatrix} \alpha(|w_0\cdots w_{i-1}|) \\ \beta(|w_0\cdots w_{i-1}|) \end{pmatrix} \cdots \begin{pmatrix} \alpha(|w_0\cdots w_i| - 1) \\ \beta(|w_0\cdots w_i| - 1) \end{pmatrix} \ ,
$$

starting in q_{i-1}', and that $S_i = [x_i]$. Note that the length of the x_i is bounded, i.e., we have $\sup_i |x_i| \le f(0) + 2^{n^2}$.

As w is consistent with a winning strategy for Player O, the run of $\mathcal{A}$ on $w = x_0x_1x_2\cdots$ is accepting. Thus, we conclude that the play $r_0q_0r_1q_1r_2q_2\cdots$ is winning for Player O, as the x_i are a bounded sequence of representatives. Hence, τ_O' is indeed a winning strategy for Player O in $\mathcal{G}$.

Now, we consider the other implication: Assume Player O has a winning strategy τ_O' for $\mathcal{G}$ and fix $d = 2^{n^2}$. We construct a winning strategy τ_O for her in $\Gamma_f(L(\mathcal{A}))$ for the constant delay function f with $f(0) = 2d$. In the following, both players pick their moves in blocks of length d. We denote Player I's blocks by a_i and Player O's blocks by b_i, i.e., in the following, every a_i is in Σ_I^d and every b_i is in Σ_O^d. This time, we simulate a play of Γ by a play in $\mathcal{G}$.

Let a_0a_1 be the first move of Player I in Γ, let $q_0 = q_I^{\mathcal{P}}$, and define the functions $r_0 = r_{\{q_0\}}(a_0)$ and $r_1 = r_{r_0(q_0)}(a_1)$ (recall the definition of r_D below Lemma 4.3). Then, $r_0q_0r_1$ is a legal play prefix of $\mathcal{G}$ that is consistent with the winning strategy τ_O' for Player O.

Thus, we are in the following situation for $i = 1$: 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'; in Γ, Player I has picked $a_0\cdots a_i$ such that a_j is a witness for r_j for every j in the range $0 \le j \le i$. Player O has picked $b_0\cdots b_{i-2}$, which is the empty word for $i = 1$.

In this situation, let $q_i = \tau_O'(r_0q_0\cdots r_{i-1}q_{i-1}r_i)$. By definition, we have $q_i \in \operatorname{dom}(r_i) = r_{i-1}(q_{i-1})$. Furthermore, as a_{i-1} is a witness for r_{i-1}, there exists b_{i-1} such that $\mathcal{P}$ reaches the state q_i when processing $\binom{a_{i-1}}{b_{i-1}}$ starting in state $(q_{i-1}', [\varepsilon])$, where $q_{i-1} = (q_{i-1}', S_{i-1})$.

Player O's strategy for Γ is to play b_{i-1} in the next d rounds, which is answered by Player I by picking some a_{i+1} during these rounds. This induces the function $r_{i+1} = r_{r_i(q_i)}(a_{i+1})$. Now, we are in the same situation as described in the previous paragraph. This finishes the description of the strategy τ_O.

It remains to show that τ_O is winning for Player O in Γ. Let $w = \binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots$ be the outcome of a play in Γ that is consistent with τ_O. Furthermore, let $r_0q_0r_1q_1r_2q_2\cdots$ be the corresponding play in $\mathcal{G}$ constructed in the simulation as described above, which is consistent with τ_O'. Let $q_i = (q_i', S_i)$. A straightforward inductive application of Remark 4.2 shows that q_i' is the state reached by $\mathcal{A}$ after processing $x_i = \binom{a_i}{b_i}$ starting in q_{i-1}' and $S_i = [x_i]$. Furthermore, $\sup_i |x_i| = d$.

As $r_0 q_0 r_1 q_1 r_2 q_2 \cdots$ is consistent with a winning strategy for Player O and therefore winning for Player O, we conclude that $x_0 x_1 x_2 \cdots$ is accepted by $\mathcal{A}$. Hence, $\mathcal{A}$ accepts the outcome w, which is equal to $x_0 x_1 x_2 \cdots$, i.e., the play in Γ is winning for Player O. Thus, τ_O is a winning strategy for Player O in Γ. □

Now, we can prove our main theorem of this section, Theorem 4.1.

Proof. Due to Lemma 4.4, we just have to show that we can construct and solve an explicit version of $\mathcal{G}(\mathcal{A})$. First, we show how to determine $\mathfrak{R}$. The automaton $\mathcal{P}$ can be constructed by building the tracking automaton $\mathcal{T}_1$. Then, for every partial function r from $Q_{\mathcal{P}}$ to $2^{Q_{\mathcal{P}}}$ construct the automaton recognizing the language W_r of witnesses of r as described in the proof of Lemma 4.3. Then, $r \in \mathfrak{R}$ if, and only if, W_r is infinite, which can easily be checked.

Now, we encode $\mathcal{G}(\mathcal{A})$ as a graph-based game with arena (V, V_I, V_O, E) where

- the set of vertices is $V = V_I \cup V_O$ with
- the vertices $V_I = \{v_I\} \cup \mathfrak{R} \times Q_{\mathcal{P}}$ of Player I, where v_I is a fresh initial vertex,
- the vertices $V_O = \mathfrak{R}$ of Player O, and
- E is the union of the following sets of edges:
    - $\{(v_I, r) \mid \mathrm{dom}(r) = \{q_I^{\mathcal{P}}\}\}$, the initial moves of Player I,
    - $\{((r, q), r') \mid \mathrm{dom}(r') = r(q)\}$, (regular) moves of Player I, and
    - $\{(r, (r, q)) \mid q \in \mathrm{dom}(r)\}$, moves of Player O.

A play is an infinite path starting in v_I. To determine the winner of a play, we fix an arbitrary function rep: $(\Sigma_I \times \Sigma_O)^* / \equiv_{\mathcal{A}}^1 \to (\Sigma_I \times \Sigma_O)^*$ that maps each equivalence class to some representative, i.e., $\mathrm{rep}(S) \in S$ for every $S \in (\Sigma_I \times \Sigma_O)^* / \equiv_{\mathcal{A}}^1$. This can be effectively done by picking a word that leads to each reachable state of $\mathcal{T}$, as these states correspond to equivalence classes of $\equiv_{\mathcal{A}}^1$.

Now, consider an infinite play

$$v_I, r_0, (r_0, q_0), r_1, (r_1, q_1), r_2, (r_2, q_2), \ldots \ ,$$

with $q_i = (q_i', S_i)$ for every i. This play is winning for Player O, if the infinite word $\mathrm{rep}(S_0)\mathrm{rep}(S_1)\mathrm{rep}(S_2)\cdots$ is accepted by $\mathcal{A}$ (note that $\sup_i |\mathrm{rep}(S_i)|$ is bounded, as there are only finitely many equivalence classes). The set Win $\subseteq V^\omega$ of winning plays for Player O is a max-regular language[6], as it can be recognized by an automaton that simulates the run of $\mathcal{A}$ on $\mathrm{rep}(S)$ when processing a vertex of the form $(r, (q, S))$ and ignores all other vertices. Games in finite arenas with max-regular winning condition are decidable via an encoding as a satisfiability problem for WMSO+UP [3].

Player O wins $\mathcal{G}(\mathcal{A})$ (and thus $\Gamma_f(L(\mathcal{A}))$ for some constant f) if, and only if, she has a winning strategy from v_I in the game $((V, V_I, V_O, E), \mathrm{Win})$. □

We obtain a doubly-exponential upper bound on the constant lookahead necessary for Player O to win a delay game with a max-regular winning condition by applying both directions of the equivalence between $\Gamma_f(L(\mathcal{A}))$ and $\mathcal{G}(\mathcal{A})$: If

---

[6]This implies that $\mathcal{G}(\mathcal{A})$ is determined, as max-regular conditions are Borel [2].

Player O wins $\Gamma_f(L(\mathcal{A}))$ for some constant f, then she also wins $\mathcal{G}(\mathcal{A})$, and therefore also $\Gamma_f(L(\mathcal{A}))$ for the constant f with $f(0) = 2d$, where d is defined as in the proof of Lemma 4.4.

Corollary 4.5. *Let $\mathcal{A}$ be a max-automaton with n states and k counters. The following are equivalent:*

(1) *Player O wins $\Gamma_f(L(\mathcal{A}))$ for some constant delay function f.*
(2) *Player O wins $\Gamma_f(L(\mathcal{A}))$ for some constant delay function f with*
   $$f(0) \le 2^{n^2 \cdot 2^{2n(\log(n)+4(k^2+k))}+1}.$$

5. BOUNDED LOOKAHEAD DOES NOT SUFFICE

In the previous section, we proved that the winner of a max-regular delay game with respect to constant delay functions can be determined effectively. However, in this section, we show that bounded and thus constant lookahead does not suffice to win every max-regular delay game that Player O can win with arbitrary lookahead. Thus, in this aspect, the max-regular languages behave differently than the ω-regular ones.

Theorem 5.1. *There is a max-regular language L such that Player O wins $\Gamma_f(L)$ for every unbounded delay function f, but not for any bounded delay function f.*

Proof. Let $\Sigma_I = \{0, 1, \#\}$ and $\Sigma_O = \{0, 1, *\}$. An input block is a word $\#w$ with $w \in \{0, 1\}^+$. An output block is a word $\binom{\#}{\alpha(n)}\binom{\alpha(1)}{*}\binom{\alpha(2)}{*}\cdots\binom{\alpha(n-1)}{*}\binom{\alpha(n)}{\alpha(n)} \in (\Sigma_I \times \Sigma_O)^+$ with $\alpha(j) \in \{0, 1\}$ for all j in the range $1 \le j \le n$. The first and last letter in an output block are the only ones whose second component is not an $*$, and these bits have to be equal to the first component of the block's last letter. Every input block of length n can be extended to an output block of length n and projecting an output block to its first components yields an input block.

Let $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$ be the language of words $\binom{\alpha}{\beta}$ satisfying the following property: If α contains infinitely many $\#$ and arbitrarily long input blocks, then $\binom{\alpha}{\beta}$ contains arbitrarily long output blocks (note that we do *not* require the output blocks to be maximal in the sense that they end just before a position where Player I has picked a $\#$). It is easy to come up with a WMSO+U formula defining L by formalizing the definitions of input and output blocks in first-order logic.

Now, consider L as winning condition for a delay game. Intuitively, Player O has to specify arbitrarily long output blocks, provided Player I produces arbitrarily long input blocks. The challenge for Player O is that she has to specify at the beginning of every output block whether she ends the block in a position where Player I has picked a 0 or a 1.

First, consider $\Gamma_f(L)$ for an unbounded delay function f. The following strategy is winning for Player O: Whenever she has to pick $\beta(i)$ at a position where Player I picked $\alpha(i) = \#$, she picks the last letter of the longest input block in the lookahead that starts with the current $\#$. Then, she completes the output block by picking $*$ until the end of the input block, where she copies $\beta(i)$, which completes the

output block. At every other position, she picks an arbitrary letter. Now, consider a play consistent with this strategy: If Player I picks infinitely many # and arbitrarily large input blocks, then Player O sees arbitrarily large input blocks in her lookahead, i.e., her strategy picks arbitrarily large output blocks. Thus, the strategy is indeed winning.

It remains to show that Player I wins $\Gamma_f(L)$ for every bounded delay function f. Due to Lemma 2.2, it suffices to only consider constant delay functions.

Hence, fix such a function and define $\ell = f(0)$, i.e., ℓ is the size of the lookahead Player O has in each round. Player I produces longer and longer input blocks of the following form: He starts picking # followed by 0's until Player O has picked an answer at the position of the last #. If she picked a 0, then Player I finishes the input block by picking 1's; if she picked a 1 (or an $*$), then he finishes the input block by picking 0's. Thus, the length of every output block is at most ℓ, since Player O has to determine the answer to every # after seeing the the next ℓ letters picked by Player I. Thus, Player I picks infinitely many # and arbitrarily long input blocks, while the length of the output blocks is bounded. Hence, the strategy is winning for Player I. □

6. Max-regular Delay Games with Unbounded Lookahead

In this section, we complement the result of the previous section, showing that bounded lookahead is not always sufficient for max-regular delay games, by showing that any unbounded lookahead is sufficient for Player O, provided some lookahead allows her to win at all. It is easy to see that if Player O wins a game with respect to some delay function f, then she also wins with respect to every f' that grants her at every round larger lookahead. The hard part of the proof is to show that she also wins for *smaller* functions f' that grant her less lookahead.

We show this by defining another game $\mathcal{G}'(\mathcal{A})$ based on the equivalence relations capturing the behavior of max-automata. This time, as we have to deal with unbounded delay functions, we use the relations $\equiv_{\mathcal{A}}^m$ for arbitrarily large m: Player I picks equivalence classes of $=_{\mathcal{A}}^m$ for increasing m and Player O picks compatible $\equiv_{\mathcal{A}}^m$ classes. The rate of m's convergence to infinity is controlled by Player I. In particular, he loses if m does not tend to infinity. If it does, then by picking representatives of the $\equiv_{\mathcal{A}}^m$ classes picked by Player O, one obtains a word whose membership in $L(\mathcal{A})$ determines the winner. Also, Player I is always one move ahead to account for the delay. This game allows to prove that smaller, but unbounded, lookahead is also sufficient, as Player I is in charge of the precision and may increase it as slowly as he wants to.

Theorem 6.1. *Let $\mathcal{A}$ be a max-automaton with n states and k counters and let $d = 2^{2^{n(\log(n)+2(k^2+k))}}$. The following are equivalent:*

 (1) Player O wins $\Gamma_f(L(\mathcal{A}))$ for some f.
 (2) Player O wins $\Gamma_f(L(\mathcal{A}))$ for every unbounded f with $f(0) \geq 2d$.

Fix $\mathcal{A} = (Q, C, \Sigma_I \times \Sigma_O, q_I, \delta, \ell, \varphi)$ with $|Q| = n$ and $|C| = k$. We define the game $\mathcal{G}'(\mathcal{A})$ between Player I and Player O played in rounds $i = 0, 1, 2, \dots$

as follows: In round 0, Player I picks natural numbers r_0, r_1 and picks infinite equivalence classes $[x_0]_{=_\mathcal{A}^{r_0}}$ and $[x_1]_{=_\mathcal{A}^{r_1}}$. Then, Player O picks an equivalence class $[\binom{x_0}{y_0}]_{\equiv_\mathcal{A}^{r_0}}$. Note that this choice is independent of the representative x_0. Now, consider round $i > 0$: Player I picks $r_{i+1} \in \mathbb{N}$ and an infinite equivalence class $[x_{i+1}]_{=_\mathcal{A}^{r_{i+1}}}$. Afterwards, Player O picks an equivalence class $[\binom{x_i}{y_i}]_{\equiv_\mathcal{A}^{r_i}}$, whose choice is again independent of the representative x_i.

Thus, the players produce a play

$$[x_0]_{=_\mathcal{A}^{r_0}} \; [\binom{x_0}{y_0}]_{\equiv_\mathcal{A}^{r_0}} \; [x_1]_{=_\mathcal{A}^{r_1}} \; [\binom{x_1}{y_1}]_{\equiv_\mathcal{A}^{r_1}} \; [x_2]_{=_\mathcal{A}^{r_2}} \; [\binom{x_2}{y_2}]_{\equiv_\mathcal{A}^{r_2}} \cdots$$

(note that this does not represent the order in which the players made their moves). Player O wins, if $(r_i)_{i \in \mathbb{N}}$ is not a rate or if $\binom{x_0}{y_0}\binom{x_1}{y_1}\binom{x_2}{y_2} \cdots \in L(\mathcal{A})$. Otherwise, i.e., if $(r_i)_{i \in \mathbb{N}}$ is a rate and $\binom{x_0}{y_0}\binom{x_1}{y_1}\binom{x_2}{y_2} \cdots \notin L(\mathcal{A})$, Player I wins. By Item (2) of Lemma 3.3, winning does not depend on the choice of representatives x_i and y_i. Strategies and winning strategies for $\mathcal{G}'(\mathcal{A})$ are defined as expected, taking into account that Player I is always one equivalence class ahead.

The following lemma about the relation between $\Gamma_f(L(\mathcal{A}))$ and $\mathcal{G}'(\mathcal{A})$ implies Theorem 6.1.

Lemma 6.2. *The following are equivalent:*

(1) *Player O wins $\Gamma_f(L(\mathcal{A}))$ for some f.*
(2) *Player O wins $\Gamma_f(L(\mathcal{A}))$ for every unbounded f with $f(0) \geq 2d$.*
(3) *Player O wins $\mathcal{G}'(\mathcal{A})$.*

Proof. It suffices to show that (1) implies (3) and that (3) implies (2), as (2) implies (1) is trivially true. For the sake of readability, we write Γ instead of $\Gamma_f(L(\mathcal{A}))$, as long as f is clear from context. Similarly, we write $\mathcal{G}'$ instead of $\mathcal{G}'(\mathcal{A})$.

Let Player O win $\Gamma_f(L(\mathcal{A}))$ for some f, say with winning strategy τ_O. We construct a winning strategy τ'_O for her in $\mathcal{G}'$ by simulating a play in Γ that is consistent with τ_O.

In round 0 of $\mathcal{G}'$, Player I picks r_0, r_1, $[x_0]_{=_\mathcal{A}^{r_0}}$, and $[x_1]_{=_\mathcal{A}^{r_1}}$. As both equivalence classes are infinite, we can assume without loss of generality $|x_0| \geq f(0)$ and $|x_1| \geq \sum_{j=1}^{|x_0|-1} f(j)$. Now, assume Player I picks in Γ the prefix of $x_0 x_1$ of length $\sum_{j=0}^{|x_0|-1} f(j)$ during the first $|x_0|$ rounds. Let y_0 of length $|x_0|$ be the answer of Player O to these choices determined by the winning strategy τ_O. We define τ'_O such that it picks $[\binom{x_0}{y_0}]_{\equiv_\mathcal{A}^{r_0}}$ as answer to Player I picking r_0, r_1, $[x_0]_{=_\mathcal{A}^{r_0}}$, and $[x_1]_{=_\mathcal{A}^{r_1}}$ in round 0.

Now, we are in the following situation for $i = 1$: In $\mathcal{G}'$, Player I has picked natural numbers $r_0, \ldots, r_i$ and equivalence classes $[x_0]_{=_\mathcal{A}^{r_0}}, \ldots, [x_i]_{=_\mathcal{A}^{r_i}}$ such that $|x_0| \geq f(0)$, $|x_1| \geq \sum_{j=1}^{|x_0|-1} f(j)$, and

$$|x_{i'}| \geq \sum_{j=0}^{|x_{i'-1}|-1} f(|x_0 \cdots x_{i'-1}| + j)$$

for every i' with $1 < i' \leq i$ (this statement is vacuously true for $i = 1$). Player O has picked $[\binom{x_0}{y_0}]_{\equiv_{\mathcal{A}}^{r_0}}, \ldots, [\binom{x_{i-1}}{y_{i-1}}]_{\equiv_{\mathcal{A}}^{r_{i-1}}}$. Further, in Γ, Player I has picked the prefix of $x_0 \cdots x_i$ of length $\sum_{j=0}^{|x_0 \cdots x_{i-1}|-1} f(j)$ during the first $|x_0 \cdots x_{i-1}|$ rounds, which was answered by Player O according to τ_O by picking $y_0 \cdots y_{i-1}$.

In this situation, it is Player I's turn in $\mathcal{G}'$, i.e., he picks r_{i+1} and $[x_{i+1}]_{=_{\mathcal{A}}^{r_{i+1}}}$. Again, as the class is infinite, we can assume $|x_{i+1}| \geq \sum_{j=0}^{|x_i|-1} f(|x_0 \cdots x_i| + j)$. Thus, we continue the play in Γ by letting Player I pick letters such that he has picked the prefix of $x_0 \cdots x_{i+1}$ of length $\sum_{j=0}^{|x_0 \cdots x_i|-1} f(j)$ during the first $|x_0 \cdots x_i|$ rounds. Again, this is answered by Player I by picking $y_0 \cdots y_i$ such that $|y_i| = |x_i|$ according to τ_O. Now, we define τ_O' such that it picks $[\binom{x_i}{y_i}]_{\equiv_{\mathcal{A}}^{r_i}}$ as next move. Thus, we are in the situation described above for $i + 1$.

Let $w' = [x_0]_{=_{\mathcal{A}}^{r_0}} [\binom{x_0}{y_0}]_{\equiv_{\mathcal{A}}^{r_0}} [x_1]_{=_{\mathcal{A}}^{r_1}} [\binom{x_1}{y_1}]_{\equiv_{\mathcal{A}}^{r_1}} [x_2]_{=_{\mathcal{A}}^{r_2}} [\binom{x_2}{y_2}]_{\equiv_{\mathcal{A}}^{r_2}} \cdots$ be a play in $\mathcal{G}'$ that is consistent with τ_O'. Consider the outcome $w = \binom{x_0}{y_0} \binom{x_1}{y_1} \binom{x_2}{y_2} \cdots$ of the play in Γ constructed during the simulation. It is consistent with τ_O, hence $w \in L(\mathcal{A})$. Accordingly, Player O wins the play w'. Thus, τ_O' is indeed a winning strategy for Player O in $\mathcal{G}'$.

Now, consider the second implication to be proven: Assume Player O has a winning strategy τ_O' for $\mathcal{G}'$ and let f be an arbitrary unbounded delay function with $f(0) \geq 2d$. We construct a winning strategy τ_O for Player O in Γ by simulating a play of Γ in $\mathcal{G}'$.

To this end, we define a strictly increasing auxiliary rate $(d_i)_{i \in \mathbb{N}}$ recursively as follows: Let d_0 be minimal with the property that every word of length at least d_0 is in some infinite equivalence class of $=_{\mathcal{A}}^0$. We have $d_0 \leq d = 2^{2^{n(\log(n)+2(k^2+k))}}$ due to Remark 3.6. Now, we define d_{i+1} to be the minimal integer strictly greater than d_i such that every word of length at least d_{i+1} is in some infinite equivalence class of $=_{\mathcal{A}}^{i+1}$. This is well-defined due to $=_{\mathcal{A}}^{i+1}$ having finite index, i.e., there are only finitely many words in finite equivalence classes.

Let Player I pick $x_0 x_1$ of length $f(0) \geq 2 \cdot d_0$ in round 0 of Γ (the exact decomposition into x_0 and x_1 is irrelevant, we just use it to keep the notation consistent). Now, decompose $x_0 x_1 = x_0' x_1' \beta_1$ such that $|x_0'| = |x_1'| = d_0$. We simulate these moves by letting Player I pick $r_0 = r_1 = 0$, $[x_0']_{=_{\mathcal{A}}^{r_0}}$, and $[x_1']_{=_{\mathcal{A}}^{r_1}}$ in round 0 of $\mathcal{G}'$, which are legal moves by the choice of d_0.

Thus, we are in the following situation for $i = 1$: In Γ, Player I has picked $x_0 \cdots x_i$ and Player O has picked $y_0 \cdots y_{i-2}$. Furthermore, in $\mathcal{G}'$, Player I has picked $[x_0']_{=_{\mathcal{A}}^{r_0}}, \ldots, [x_i']_{=_{\mathcal{A}}^{r_i}}$ and there is a buffer $\beta_i \in \Sigma_I^*$ such that $x_0 \cdots x_i = x_0' \cdots x_i' \beta_i$. Finally, Player O has picked $[\binom{x_0'}{y_0}]_{\equiv_{\mathcal{A}}^{r_0}} \cdots [\binom{x_{i-2}'}{y_{i-2}}]_{\equiv_{\mathcal{A}}^{r_{i-2}}}$.

In this situation, it is Player O's turn and τ_O' returns a class $[\binom{x_{i-1}'}{y_{i-1}}]_{\equiv_{\mathcal{A}}^{r_{i-1}}}$. Thus, we define τ_O such that it picks y_{i-1} during the next rounds, in which Player I picks letters forming x_{i+1} satisfying $|x_{i+1}| \geq |y_{i-1}|$. We consider two cases to simulate these in $\mathcal{G}'$:

    (1) If $|\beta_i x_{i+1}| \geq 2d_{r_i+1} - d_{r_i}$, then Player I picks $r_{i+1} = r_i + 1$ and $[x_{i+1}']_{=_{\mathcal{A}}^{r_{i+1}}}$, where x_{i+1}' is the prefix of $\beta_i x_{i+1}$ of length $d_{r_{i+1}}$. This is an infinite

equivalence class by the choice of $d_{r_{i+1}}$. The remaining suffix of $\beta_i x_{i+1}$ is stored in the buffer β_{i+1}, i.e., we have $\beta_i x_{i+1} = x'_{i+1}\beta_{i+1}$.

(2) Now, consider the case $|\beta_i x_{i+1}| < 2d_{r_i+1} - d_{r_i}$: Below, we show $|\beta_i x_{i+1}| \geq d_{r_i}$. Then, Player I picks $r_{i+1} = r_i$ and $[x'_{i+1}]_{=_{\mathcal{A}}^{r_{i+1}}}$, where x'_{i+1} is the prefix of $\beta_i x_{i+1}$ of length $d_{r_{i+1}} = d_{r_i}$, which is again an infinite equivalence class by the choice of $d_{r_{i+1}}$. The remaining suffix of $\beta_i x_{i+1}$ is stored in the buffer β_{i+1}, i.e., we have $\beta_i x_{i+1} = x'_{i+1}\beta_{i+1}$.

To show $|\beta_i x_{i+1}| \geq d_{r_i}$, we again consider two cases: If $r_{i-1} = r_i$, then we have

$$|\beta_i x_{i+1}| \geq |x_{i+1}| \geq |y_{i-1}| = |x'_{i-1}| = d_{r_{i-1}} = d_{r_i}.$$

On the other hand, assume $r_{i-1} < r_i$, which implies $r_{i-1} + 1 = r_i$, as we are in case (1) of the (outer) case distinction. Then, we have $|\beta_{i-1} x_i| \geq 2d_{r_{i-1}+1} - d_{r_{i-1}}$ and x'_i is the prefix of length $d_{r_i} = d_{r_{i-1}+1}$ of $\beta_{i-1} x_i$, which implies $|\beta_i| \geq d_{r_{i-1}+1} - d_{r_{i-1}}$, as it is the remaining suffix of $\beta_{i-1} x_i$. Finally, we have

$$|x_{i+1}| \geq |y_{i-1}| = |x'_{i-1}| = d_{r_{i-1}}.$$

Altogether, we obtain

$$|\beta_i x_{i+1}| \geq (d_{r_{i-1}+1} - d_{r_{i-1}}) + d_{r_{i-1}} = d_{r_{i-1}+1} = d_{r_i}.$$

In both cases, we are back in the situation described above for $i + 1$.

Let $w = \binom{x_0 x_1 x_2 \cdots}{y_0 y_1 y_2 \cdots}$ be the outcome of a play in Γ that is consistent with τ_O. The play $[x'_0]_{=_{\mathcal{A}}^{r_0}} [\binom{x'_0}{y_0}]_{\equiv_{\mathcal{A}}^{r_0}} [x'_1]_{=_{\mathcal{A}}^{r_1}} [\binom{x'_1}{y_1}]_{\equiv_{\mathcal{A}}^{r_1}} [x'_2]_{=_{\mathcal{A}}^{r_2}} [\binom{x'_2}{y_2}]_{\equiv_{\mathcal{A}}^{r_2}} \cdots$ in $\mathcal{G}'$ constructed during the simulation is consistent with τ'_O. As f is unbounded, $(r_i)_{i\in\mathbb{N}}$ is unbounded as well and thus a rate. Hence, we conclude $\binom{x'_0}{y_0}\binom{x'_1}{y_1}\binom{x'_2}{y_2} \in L(\mathcal{A})$, as τ'_O is a winning strategy. Also, a straightforward induction shows $x_0 x_1 x_2 \cdots = x'_0 x'_1 x'_2 \cdots$. Thus, $w \in L(\mathcal{A})$, i.e., τ_O is a winning strategy for Player O in Γ. $\square$

7. Towards Solving Max-regular Delay Games with Unbounded Lookahead

Unlike for ω-regular delay games, bounded lookahead is not always sufficient for Player O to win a max-regular delay game. Hence, determining the winner with respect to arbitrary delay functions is not equivalent to determining the winner with respect to bounded delay functions, which we have shown to be decidable in Section 4. We refer to the former problem as "solving max-regular delay games". In this section, we discuss some obstacles one has to overcome in order to extend the decidability result for bounded lookahead to unbounded lookahead. Furthermore, we give straightforward lower bounds on the complexity.

Proving upper bounds, e.g., decidability of determining the winner of max-regular delay games with respect to arbitrary delay functions, is complicated by the need for unbounded lookahead. All known decidability results [20,24,26], including the one presented here, are for the case where bounded lookahead is sufficient and proceed by solving this restricted problem. in particular, the decidability proof presented here is based on the fact that the error introduced by using the imprecise equivalence relation $\equiv^1_{\mathcal{A}}$ is bounded in the context of bounded lookahead.

However, for unbounded lookahead, the error is unbounded as well. In particular, the example presented in Section 4 shows that bounded counters might grow arbitrarily large during different plays: The winning condition L described in the proof of Theorem 5.1 is recognized by a max-automaton with four counters: c_i counts the length of input blocks and is reset at every $\#$, c'_o is incremented during prefixes of possible output blocks and reset at the end of such a block. Furthermore, the value of c'_o is copied to c_o every time the requirement on the first and last letter of an output block is met. Finally, a counter $c_{\#}$ counts the number of $\#$'s in the word. The acceptance condition of the automaton recognizing L is given by the formula

$$\text{``}\limsup \rho_{c_{\#}} < \infty\text{''} \ \lor \ \text{``}\limsup \rho_{c_i} < \infty\text{''} \ \lor \ \text{``}\limsup \rho_{c_o} = \infty\text{''}.$$

As already argued, Player O has a winning strategy for $\Gamma_f(L)$, provided f is unbounded. However, she does not have a strategy that bounds the counters $c_{\#}$ and c_i to some fixed value among all consistent plays that are won due to $c_{\#}$ or c_i being bounded: For example, Player I can pick any finite number of $\#$'s and then stop doing so. This implies that $c_{\#}$ is bounded, but with an arbitrarily large value among different plays. The lack of such a uniform bound in itself is not surprising, but entails that one has to deal with arbitrarily large counter values when trying to extend the approach described above for the setting with bounded lookahead. In particular, it is not enough to replace $=^1_{\mathcal{A}}$ and $\equiv^1_{\mathcal{A}}$ by $=^m_{\mathcal{A}}$ and $\equiv^m_{\mathcal{A}}$ for some fixed m that only depends on the winning condition.

Two other possible approaches follow from the results proven in this paper: First, one could show that $\mathcal{G}'(\mathcal{A})$ can be solved effectively. However, the game is of infinite size and not in one of the classes of effectively solvable games with infinite state space, e.g., pushdown games. Second, one can pick any unbounded delay function f with large enough $f(0)$ and solve $\Gamma_f(L(\mathcal{A}))$, as winning with respect to one such function is equivalent to winning with respect to all of them. However, $\Gamma_f(L(\mathcal{A}))$ is again of infinite size and not in one of the classes of effectively solvable games with infinite state space.

One obvious reason we fail to find an algorithm solving max-regular delay games might be that the problem is undecidable. There is a class of winning conditions for which solving delay games is indeed known to be undecidable, namely (very restricted fragments of) ω-context-free conditions [18]. However, this result is based on the language $\{a^n b^n \mid n \in \mathbb{N}\}$ being context-free, which suffices to encode two-counter machines. As max-automata have no mechanism to compare arbitrarily

large numbers exactly, this simple encoding of two-counter machines cannot be captured in a delay game with max-regular winning condition.

This can be overcome by allowing quantification over arbitrary sets: Recently, and after being an open problem for more than a decade, satisfiability of MSO+U over infinite words was shown to be undecidable [6] by capturing termination of two-counter machines by MSO+U formulas based on a specially tailored encoding. However, the resulting formulas have six alternations between existential and universal set quantifiers and then a block of (negated) unbounding quantifiers. To adapt this proof to show undecidability of max-regular delay games with respect to arbitrary delay functions, one has to replace the set quantifiers by the interaction between the players, which seems unlikely to be achievable.

On the other hand, one can prove some straightforward lower bounds. As usual, solving delay games is at least as hard as solving the universality problem for the class of automata used to specify the winning conditions: Given such an automaton $\mathcal{A}$ over some alphabet Σ, we change the alphabet to $\Sigma \times \Sigma$ by replacing each letter a on a transition by the letter $\binom{a}{a}$ and route all missing transitions to a fresh rejecting sink state. Call the resulting automaton $\mathcal{A}'$. The game $\Gamma_f(\mathcal{A}')$ is won by Player O if, and only if, $L(\mathcal{A})$ is universal, independently of f: If $L(\mathcal{A})$ is not universal, then Player I can produce some $\alpha \notin L(\mathcal{A})$ and thereby win; if it is indeed universal, then Player O can mimic the choices of Player I and wins.

Proposition 7.1. *Solving max-regular delay games is at least as hard as solving the universality problem for max-automata.*

The best known lower bound on the universality problem for max-automata is PSpace-hardness, which stems from max-automata being closed under complementation and the emptiness problem being PSpace-hard [7]. The exact complexity of the emptiness problem for max-automata is, to the best of our knowledge, an open problem.

Another lower bound is obtained by considering delay games with weaker winning conditions: Solving delay games with winning conditions recognized by deterministic safety automata is ExpTime-complete [24]. Such automata can be transformed into max-automata without increasing the number of states: Turn the non-safe states into sinks and increment a designated counter c on every transition not leading into a non-safe state. Then, the max-automaton with acceptance condition "$\limsup \rho_c = \infty$" recognizes the same language as the original safety automaton. Hence, we obtain the following lower bound.

Theorem 7.2. *Solving max-regular delay games is* ExpTime-*hard.*

This lower bound is oblivious to the intricate acceptance condition of max-automata and relies solely on the transition structure. This is in line with results for ω-regular games: Solving delay games with winning conditions given by deterministic parity automata is in ExpTime, i.e., it matches the lower bound for the special case of safety. It is open whether moving to more concise acceptance conditions for deterministic ω-automata, e.g., Rabin, Streett, and Muller, increases the complexity. These results would directly transfer to max-automata as well. Another aspect that is not exploited by this reduction is the unbounded lookahead:

The safety delay game is always winnable with bounded lookahead. We are currently investigating whether these aspects can be exploited to improve the bounds.

8. Conclusion

We considered delay games with max-regular winning conditions. Our main result is an algorithm that determines whether Player O has a winning strategy for some constant delay function, which consists of reducing the original problem to a delay-free game with max-regular winning condition. Such a game can be solved by encoding it as an emptiness problem for a certain class of tree automata (so-called WMSO+UP automata) that capture WMSO+UP on infinite trees. Our reduction also yields a doubly-exponential upper bound on the necessary constant lookahead to win such a game, provided Player O does win for some constant delay function. It is open whether the doubly-exponential upper bound is tight. The best lower bounds are exponential and hold already for deterministic reachability and safety automata [24], which can easily be transformed into max-automata.

We deliberately skipped the complexity analysis of our algorithm, since the reduction of the delay-free game to an emptiness problem for WMSO+UP automata does most likely not yield tight upper bounds on the complexity. Instead, we propose to investigate (delay-free) games with max-regular winning conditions, a problem that is worthwhile studying on its own, and to find a direct solution algorithm. Currently, the best lower bound on the computational complexity of determining whether Player O wins a delay game with max-regular winning condition for some constant delay function is the ExpTime-hardness result for games with safety conditions [24].

Also, we showed that constant lookahead is not sufficient for max-regular conditions by giving a max-regular winning condition L such that Player O wins $\Gamma_f(L)$ for every unbounded f, but not for any bounded delay function f.

Both the lower bound on the necessary lookahead and the one on the computational complexity for safety conditions mentioned above are complemented by matching upper bounds for games with parity conditions [24], i.e., having a parity condition instead of a safety condition has no discernible influence. Stated differently, the complexity of the problems manifests itself in the transition structure of the automaton. Our example from Section 5 shows that this is no longer true for max-regular conditions: Having a quantitative acceptance condition requires growing lookahead.

Finally, we showed that even though max-regular winning conditions require in general unbounded lookahead, they cannot enforce any lower bound on the growth, unlike ω-context-free conditions. In ongoing work, we aim to solve delay games with respect to arbitrary delay functions.

Due to the need for unbounded lookahead and the need for solving WMSO+UP satisfiability for solving max-regular delay games, we are currently investigating tractable quantitative fragments of WMSO+U. In preliminary work, we have shown that Prompt-LTL has better properties [26]: Triply-exponential constant

lookahead is always sufficient (and in general necessary) and solving delay games with Prompt-LTL winning conditions is complete for triply-exponential time.

As noticed by one of the reviewers, the correctness of the games $\mathcal{G}(\mathcal{A})$ and $\mathcal{G}'(\mathcal{A})$ only depends on the properties of the equivalence relations specified in Lemma 3.2 and Lemma 3.3. Hence, as soon as one can devise equivalence relations with the same properties, one obtains decidability of delay games with bounded lookahead respectively the winner being the same for every unbounded delay function.

References

[1] Mikołaj Bojańczyk. A bounding quantifier. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *CSL 2004*, volume 3210 of *LNCS*, pages 41–55. Springer, 2004.

[2] Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.

[3] Mikołaj Bojańczyk. Weak MSO+U with path quantifiers over infinite trees. In Esparza et al. [15], pages 38–49.

[4] Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS 2006*, pages 285–296. IEEE Computer Society, 2006.

[5] Mikołaj Bojańczyk, Tomasz Gogacz, Henryk Michalewski, and Michał Skrzypczak. On the decidability of MSO+U on infinite trees. In Esparza et al. [15], pages 50–61.

[6] Mikołaj Bojańczyk, Pawel Parys, and Szymon Toruńczyk. The MSO+U theory of $(\mathbb{N}, <)$ is undecidable. In Nicolas Ollinger and Heribert Vollmer, editors, *STACS 2016*, volume 47 of *LIPIcs*, pages 21:1–21:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[7] Mikołaj Bojańczyk and Szymon Toruńczyk. Deterministic automata and extensions of weak MSO. In Kannan and Kumar [23], pages 73–84.

[8] Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In Christoph Dürr and Thomas Wilke, editors, *STACS 2012*, volume 14 of *LIPIcs*, pages 648–660. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[9] J. Richard Büchi. On a decision method in restricted second-order arithmetic. In *International Congress on Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[10] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.*, 138:pp. 295–311, 1969.

[11] Jérémie Cabessa, Jacques Duparc, Alessandro Facchini, and Filip Murlak. The Wadge hierarchy of max-regular languages. In Kannan and Kumar [23], pages 121–132.

[12] Arnaud Carayol and Christof Löding. MSO on the infinite binary tree: Choice and order. In Jacques Duparc and Thomas A. Henzinger, editors, *CSL 2007*, volume 4646 of *LNCS*, pages 161–176. Springer, 2007.

[13] Arnaud Carayol and Christof Löding. Uniformization in automata theory. In Peter Schroeder-Heister, Gerhard Heinzmann, Wilfrid Hodges, and Pierre Edouard Bour, editors, *International Congress of Logic, Methodology and Philosophy of Science*. College Publications, 2015.

[14] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009.

[15] Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors. *ICALP 2014, Part II*, volume 8573 of *LNCS*. Springer, 2014.

[16] Peter Faymonville and Martin Zimmermann. Parametric linear dynamic logic. In Peron and Piazza [29], pages 60–73.

[17] Nathanaël Fijalkow and Martin Zimmermann. Parity and Streett games with costs. *LMCS*, 10(2), 2014.

[18] Wladimir Fridman, Christof Löding, and Martin Zimmermann. Degrees of lookahead in context-free infinite games. In Marc Bezem, editor, *CSL 2011*, volume 12 of *LIPIcs*, pages 264–276. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[19] Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem. *The Journal of Symbolic Logic*, 48:1105–1119, 1983.

[20] Michael Holtmann, Łukasz Kaiser, and Wolfgang Thomas. Degrees of lookahead in regular infinite games. *LMCS*, 8(3), 2012.

[21] Frederick A. Hosch and Lawrence H. Landweber. Finite delay solutions for sequential conditions. In *ICALP 1972*, pages 45–60, 1972.

[22] Szczepan Hummel and Michał Skrzypczak. The topological complexity of MSO+U and related automata models. *Fundam. Inform.*, 119(1):87–111, 2012.

[23] Ravi Kannan and K. Narayan Kumar, editors. *FSTTCS 2009*, volume 4 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009.

[24] Felix Klein and Martin Zimmermann. How much lookahead is needed to win infinite games? In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015, Part II*, volume 9135 of *LNCS*, pages 452–463. Springer, 2015.

[25] Felix Klein and Martin Zimmermann. What are strategies in delay games? borel determinacy for games with lookahead. In Stephan Kreutzer, editor, *CSL 2015*, volume 41 of *LIPIcs*, pages 519–533. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[26] Felix Klein and Martin Zimmermann. Prompt delay. *arXiv*, 1602.05045, 2016.

[27] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.

[28] Christof Löding and Sarah Winter. Synthesis of deterministic top-down tree transducers from automatic tree relations. In Peron and Piazza [29], pages 88–101.

[29] Adriano Peron and Carla Piazza, editors. *GandALF 2014*, volume 161 of *EPTCS*, 2014.

[30] Wolfgang Thomas. Infinite games and uniformization. In Mohua Banerjee and Anil Seth, editors, *ICLA*, volume 6521 of *LNCS*, pages 19–21. Springer, 2011.

[31] Wolfgang Thomas and Helmut Lescow. Logical specifications of infinite computations. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium*, volume 803 of *LNCS*, pages 583–621. Springer, 1993.

[32] Boris A. Trakhtenbrot and Ya.M. Barzdin. *Finite Automata; Behavior and Synthesis*. Fundamental Studies in Computer Science, V. 1. North-Holland Publishing Company; New York: American Elsevier, 1973.

[33] Martin Zimmermann. Optimal Bounds in Parametric LTL Games. *Theoret. Comput. Sci.*, 493(0):30 – 45, 2013.

[34] Martin Zimmermann. Delay games with WMSO+U winning conditions. In Lev D. Beklemishev and Daniil V. Musatov, editors, *CSR 2015*, volume 9139 of *LNCS*, pages 412–425. Springer, 2015.