Approximating the Minimal Lookahead Needed to Win Infinite Games

Martin Zimmermann
University of Liverpool, Liverpool L69 3BX, United Kingdom
martin.zimmermann@liverpool.ac.uk

Abstract

We present an exponential-time algorithm approximating the minimal lookahead necessary to win an ω-regular delay game.

1 Introduction

Games can be found in the standard toolkit for many areas of theoretical computer science and mathematics, e.g., set theory and logic, automata theory, and complexity theory. Here, we are concerned with two-player zero-sum games of infinite duration and perfect information. In their most abstract form, these are known as Gale-Stewart games [2], played between Player I and Player O in rounds $i \in \mathbb{N}$: In each round i, first Player I picks some letter a_i from an alphabet Σ_I, then Player O picks a letter b_i from an alphabet Σ_O. Thus, after ω rounds, they have produced an infinite sequence $w = \binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots$ of letters. Now, Player O wins such a play, if w satisfies some given winning condition, e.g., $w \in L$ for some given set L of infinite words. In this work, we only consider ω-regular L given by deterministic parity automata.

Note that here both players move in strict alternation and the automaton will process the sequence in this order. Hosch and Landweber introduced delay games, relaxing this rigid interaction by allowing Player O to delay her moves in order to obtain a lookahead on Player I's moves [5].[1]

Hosch and Landweber proved that it is decidable whether Player O wins an ω-regular delay game with some bounded lookahead. In later work, Holtmann, Kaiser, and Thomas showed that such bounded lookahead is sufficient in the following sense: in an ω-regular delay game, Player O either wins with doubly-exponential lookahead or not at all (not even with unbounded lookahead) [4]. In subsequent work, an improved exponential upper bound and matching lower bounds have been proved [6].

Here, we consider the problem of determining the minimal lookahead that is sufficient for Player O to win an ω-regular delay game. It is trivial to determine this value in doubly-exponential-time by hardcoding the exponential lookahead into the game, thereby turning the delay game into a classical, i.e., delay-free, game (see [8], Section 3.1 for details). As the resulting games can be solved in doubly-exponential time, one obtains the minimal lookahead in doubly-exponential time by exhaustive search. However, this has to be contrasted with the ExpTime-hardness of determining whether Player O wins with some lookahead [6], the only known lower bound on the complexity of the optimization problem.

We present the first improvement over the naive algorithm for the lookahead optimization problem by presenting an exponential-time algorithm approximating the minimal lookahead within a factor of two. To this end, we show that the exponential-time algorithm for determining whether Player O wins for some lookahead can be refined into an approximation algorithm with an exponential running time. Due to the hardness result for the related decision problem, this is the best running time one can hope for (barring major surprises in complexity theory).

2 ω-regular Delay Games

Given an alphabet Σ, i.e., a non-empty finite set of letters, Σ^* and Σ^ω denote the set of finite respectively infinite words over Σ. Given a product alphabet $\Sigma_I \times \Sigma_O$ we write $\binom{a_0a_1a_2\cdots}{b_0b_1b_2\cdots}$ for the word $\binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots$

[1] We refer to the introduction of [4] for a discussion of the history of delay games, including motivation and a connection to the uniformization of ω-regular relations by continous functions.

with $a_i \in \Sigma_I$ and $b_i \in \Sigma_O$. Also, we use similar notation for finite words, provided they are of the same length. We denote the empty word by ε, the power set of a set S by 2^S, and the set of non-negative integers by $\mathbb{N}$.

A delay game (with constant lookahead) $\Gamma_k(L)$ consists of a lookahead $k \in \mathbb{N}$ and a winning condition $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$. It is played in rounds $i \in \mathbb{N}$ as follows: In round 0, Player I picks letters $a_0 \cdots a_k$ from Σ_I, then Player O picks a letter b_0 from Σ_O. In round $i > 0$, Player I picks a letter $a_{k+i} \in \Sigma_I$ and then Player O picks a letter $b_i \in \Sigma_O$. After ω rounds, they have produced an outcome $\binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots$. We say that the outcome is winning for Player O if it is in L.

A strategy for Player O is a mapping $\sigma\colon \Sigma_I^* \to \Sigma_O$. An outcome $\binom{a_0}{b_0}\binom{a_1}{b_1}\binom{a_2}{b_2}\cdots$ is consistent with σ, if $b_i = \sigma(a_0 \cdots a_{i+k})$ for all i. A strategy σ is winning, if every outcome that is consistent with σ is winning for Player O. If Player O has a winning strategy for $\Gamma_k(L)$, then we say she wins $\Gamma_k(L)$.

Remark 1 ([4], Remark 3.3)**.** *If Player O wins $\Gamma_k(L)$, then she also wins $\Gamma_{k'}(L)$ for every $k' > k$.*

In this work, we consider winning conditions L recognized by deterministic parity automata (DPA) $\mathfrak{A} = (Q, \Sigma, q_\iota, \delta, \Omega)$, where Q is a finite set of states containing the initial state q_ι, Σ is the input alphabet, $\delta\colon Q \times \Sigma \to Q$ is the transition function, and $\Omega\colon Q \to \mathbb{N}$ is a coloring of the states. As usual, we extend δ to finite words by defining $\delta^*\colon Q \times \Sigma^* \to Q$ via $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$ for all $q \in Q$, $w \in \Sigma^*$, and $a \in \Sigma$. Given a word $a_0 a_1 a_2 \cdots \in \Sigma^\omega$, the run of $\mathfrak{A}$ on α is the unique sequence $q_0 q_1 q_2 \cdots$ of states given by $q_i = \delta^*(q_\iota, a_0 \cdots a_{i-1})$. A run $q_0 q_1 q_2 \cdots$ is accepting, if $\limsup_{i\to\infty} \Omega(q_i)$, i.e., the maximal color occurring infinitely often, is even. The language $L(\mathfrak{A})$ recognized by $\mathfrak{A}$ is the set containing all words whose run is accepting.

It is known that one can determine in exponential time whether Player O wins a delay game for some lookahead k, if the winning condition L is recognized by a DPA.

Proposition 1 ([6], Theorem 4.4)**.** *The following problem is* ExpTime*-complete: Given a DPA $\mathfrak{A}$, does Player O win $\Gamma_k(L(\mathfrak{A}))$ for some k?*

Furthermore, there is an exponential upper bound on the lookahead necessary to win a delay game.

Proposition 2 ([6], Theorem 4.8)**.** *Let $\mathfrak{A}$ be a DPA with n states and c colors, and define $k_{\max} = 2^{n^2 c+1}$. Player O wins $\Gamma_k(L(\mathfrak{A}))$ for some k if, and only if, she wins $\Gamma_{k_{\max}}(L(\mathfrak{A}))$.*

Finally, the exponential upper bound on the necessary lookahead is tight.

Proposition 3 ([6], Theorem 3.2)**.** *For every $n > 1$, there is a language L_n recognized by a DPA with $\mathcal{O}(n)$ states and two colors such that Player O wins $\Gamma_k(L_n)$ for some k, but she does not win $\Gamma_{2^n}(L_n)$.*

In this work, we consider the following problem: Given a DPA $\mathfrak{A}$ over $\Sigma_I \times \Sigma_O$, determine the smallest k such Player O wins $\Gamma_k(L(\mathfrak{A}))$ (or that there is no such k). Due to Proposition 2, the search space for the smallest such k is bounded by $k_{\max} = 2^{n^2 c+1}$.

Now, one can easily transform a game $\Gamma_k(L(\mathfrak{A}))$ (for some fixed k) into an equivalent classical parity game (see, e.g., [3] for an introduction to parity games) encoding a queue of k letters from Σ_I implementing the lookahead ([8], Section 3.1). Thus, one can construct the equivalent parity game for each $k \leq k_{\max}$ and determine the smallest k such that Player O wins the resulting parity game. This is also the smallest k such that Player O wins $\Gamma_k(L(\mathfrak{A}))$.

However, if k is exponential in the size of $\mathfrak{A}$ (i.e., close to $k_{\max}$), then the resulting parity game is of doubly-exponential size in the size of $\mathfrak{A}$, as one encodes a queue of exponential length. Due to Proposition 3, considering an exponential k is, in general, unavoidable. Hence, the resulting algorithm has doubly-exponential running time.

In the following, we show that the minimal lookahead can be approximated within a factor of two in exponential time. As the related decision problem is ExpTime-hard (Proposition 1), one cannot do better than exponential time (barring major surprises in complexity theory).

3 The Algorithm

Given a DPA $\mathfrak{A}$ over $\Sigma_I \times \Sigma_O$ and a $k > 0$, we define a game $\mathcal{G}_k$ played between Player I and O with the following properties:

1. If Player O wins $\Gamma_k(L(\mathfrak{A}))$, then she wins $\mathcal{G}_k$ (Lemma 1).
2. If Player O wins $\mathcal{G}_k$, then she wins $\Gamma_{2k-1}(L(\mathfrak{A}))$ (Lemma 2).
3. Given $\mathfrak{A}$ and $k \leq 2^{n^2c+1}$, one can construct $\mathcal{G}_k$ and determine its winner in exponential time in n, where n and c are the number of states and colors of $\mathfrak{A}$ (Lemma 3).

Now, consider Algorithm 1.

Algorithm 1 Approximating the minimal lookahead necessary to win a delay game with winning condition $L(\mathfrak{A})$, where $\mathfrak{A}$ is a given DPA with n states and c colors

```
1: for k = 1 to 2^{n^2 c+1} do
2:   if Player O wins G_k then
3:     return 2k − 1
4: return "Player O does not win for any lookahead k"
```

.

It is obvious that the algorithm runs in exponential time, as the calls in Line 2 can be executed in exponential time (Property 3) and the loop terminates after an exponential number of iterations (which can obviously be reduced to a polynomial number of iterations using binary search).

Now, fix an input $\mathfrak{A}$. If Player O does not win $\Gamma_k(L(\mathfrak{A}))$ for any k, then she does not win any $\mathcal{G}_k$ (Property 2), i.e., the algorithm returns the correct output in Line 4. So, consider the case where Player O wins $\Gamma_k(L(\mathfrak{A}))$ for some k. Further, let k_{opt} be the minimal k such that Player O wins $\Gamma_k(L(\mathfrak{A}))$. Proposition 2 yields $k_{\text{opt}} \leq 2^{n^2c+1}$. Hence, Player O wins $\mathcal{G}_k$ for some $k \leq 2^{n^2c+1}$ due to Property 1. We pick k^* minimal with this property, i.e., $2k^* - 1$ is the output of the algorithm. Due to Property 2, the output $2k^* - 1$ allows Player O to win the delay game with winning condition $L(\mathfrak{A})$. Finally, the algorithm indeed approximates the minimal lookahead within a factor of two: Due to Property 1, we have $k^* \leq k_{\text{opt}}$, which implies that the approximation ratio between the algorithm's output $2k^* - 1$ and the optimal value k_{opt} is indeed bounded by two:

$$\frac{2k^* - 1}{k_{\text{opt}}} \leq \frac{2k^*}{k_{\text{opt}}} \leq \frac{2k_{\text{opt}}}{k_{\text{opt}}} \leq 2.$$

Altogether, we obtain our main result.

Theorem 1. *The following problem can be approximated within a factor of two in exponential time: Given a DPA $\mathfrak{A}$, determine the smallest k such that Player O wins $\Gamma_k(L(\mathfrak{A}))$.*

Note that we do not consider the computation of a strategy realizing the approximation, as the notion of finite-state strategies for delay games comes with some technical complications [8].

In the remainder of this section, we present the construction of $\mathcal{G}_k$ and prove Properties 1, 2, and 3.

3.1 The Game $\mathcal{G}_k$

The construction of $\mathcal{G}_k$ is a refinement of a similar game used to prove Proposition 1 [6, 8]. For a detailed explanation of the construction, we refer the reader to these works.

Throughout this section, we fix $\mathfrak{A} = (Q, \Sigma_I \times \Sigma_O, q_\iota, \delta, \Omega)$, some $0 < k \leq 2^{n^2c+1}$, and let $C = \Omega(Q)$ denote the set of colors of $\mathfrak{A}$.

First, we modify the transition function of $\mathfrak{A}$ so that it keeps track of the maximal color occurring along a (partial) run of $\mathfrak{A}$. Formally, we define $\delta_{\mathcal{T}} \colon (Q \times C) \times (\Sigma_I \times \Sigma_O) \to (Q \times C)$ via

$$\delta_{\mathcal{T}}\left((q,c), \binom{a}{b}\right) = \left(\delta\left(q, \binom{a}{b}\right), \max\left\{c, \Omega\left(\delta\left(q, \binom{a}{b}\right)\right)\right\}\right)$$

for all $q \in Q$, $c \in C$, and $\binom{a}{b} \in \Sigma_I \times \Sigma_O$.

Next, we project away the Σ_O-component of the letter and perform a power set construction by defining $\delta_{\mathcal{P}} \colon 2^{Q \times C} \times \Sigma_I \to 2^{Q \times C}$ via

$$\delta_{\mathcal{P}}(S, a) = \bigcup_{(q,c) \in S} \bigcup_{b \in \Sigma_O} \delta_{\mathcal{T}}\left((q,c), \binom{a}{b}\right)$$

for all $S \subseteq Q \times C$ and $a \in \Sigma_I$. We extend $\delta_{\mathcal{P}}$ to non-empty words via $\delta^*_{\mathcal{P}}(S, \varepsilon) = S$ and $\delta^*_P(S, wa) = \delta_P(\delta^*_{\mathcal{P}}(S, w), a)$ for all $w \in \Sigma_I^*$ and $a \in \Sigma_I$.

Finally, for every non-empty $D \subseteq Q \times C$ and $w \in \Sigma_I^*$, we define the function $r_w^D \colon D \to 2^{Q \times C}$ via

$$r_w^D(q, c) = \delta^*_{\mathcal{P}}(\{(q, c)\}, w)$$

for all $(q, \Omega(q)) \in D$. Note that the first argument of $\delta^*_{\mathcal{P}}$ is $(q, \Omega(q))$ and not (q, c), the argument of r_w^D.

Remark 2. *$(q', c') \in r_w^D(q, c)$ if and only if there is a word w over $\Sigma_I \times \Sigma_O$ whose projection to Σ_I is w and such that the run of $\mathfrak{A}$ processing w from q leads to q' and has maximal color c'.*

We call $w \in \Sigma_I^k$ a witness for a partial function $r \colon Q \times C \to 2^{Q \times C}$ if we have $r = r_w^{\mathrm{dom}(r)}$, where $\mathrm{dom}(r)$ denotes the domain of r. Note that we require a witness to have length k. Let $\mathfrak{R}$ be the set of all such functions that have a witness.

Now, we define $\mathcal{G}_k$, which is played in rounds $i \in \mathbb{N}$ between Player I and Player O. In each round i, Player I has to pick some $r_i \in \mathfrak{R}$ and then Player O picks $(q_i, c_i) \in Q \times C$ subject to the following constraints:

- For Player I: $\mathrm{dom}(r_0) = \{(q_\iota, \Omega(q_\iota))\}$ and $\mathrm{dom}(r_i) = r_{i-1}(q_{i-1})$ for all $i > 0$.
- For Player O: $q_i \in \mathrm{dom}(r_i)$ for all $i \geq 0$.

It is straightforward to verify that both players always have at least one move available in every round. A play of $\mathcal{G}_k$ is a sequence $r_0(q_0, c_0)r_1(q_1, c_1)r_2(q_2, c_2)\cdots$. It is winning for Player O if the sequence of colors satisfies the parity condition, i.e., if $\limsup_{i \to \infty} c_i$ is even.

A strategy σ for Player O maps a sequence $r_0(q_0, c_0) \cdots r_i$ to a pair $(q_i, c_i) \in \mathrm{dom}(r_i)$. A play $r_0(q_0, c_0)r_1(q_1, c_1)r_2(q_2, c_2)\cdots$ is consistent with σ if $(q_i, c_i) = \sigma(r_0(q_0, c_0) \cdots r_i)$ for all $i \geq 0$. We say that σ is a winning strategy for Player O if every play that is consistent with σ is winning for her. Finally, Player O wins $\mathcal{G}_k$ if she has a winning strategy.

3.2 Correctness

Lemma 1. *If Player O wins $\Gamma_k(L(\mathfrak{A}))$, then she wins $\mathcal{G}_k$.*

Proof. Let σ be a winning strategy for Player O in $\Gamma_k(L(\mathfrak{A}))$. We construct a winning strategy σ' for Player O in $\mathcal{G}_k$, which will simulate σ.

So, let $r_0 \in \mathfrak{R}$ be the first move of Player I. This has to be answered by Player O by picking $(q_0, c_0) = (q_\iota, \Omega(q_\iota))$, as this is the only legal move for her. Hence, we define $\sigma'(r_0) = (q_\iota, \Omega(q_\iota))$. Now, Player I picks some $r_1 \in \mathfrak{R}$.

We simulate this in $\Gamma_k(L(\mathfrak{A}))$ as follows. Pick witnesses w_0 and w_1 for r_0 and r_1, respectively. If Player I uses $w_0 w_1$ during the first k rounds of $\Gamma_k(L(\mathfrak{A}))$, then σ yields k letters $w_0' \in \Sigma_O^k$ as response. Thus, we are in the following situation for $i = 1$:

- In $\mathcal{G}_k$, we have a play prefix $r_0(q_0, c_0) \cdots (q_{i-1}, c_{i-1})r_i$, and
- in $\Gamma_k(L(\mathfrak{A}))$, Player I has picked $w_0 \cdots w_i$ and Player $w_0' \cdots w_{i-1}'$, where each w_j is a witness for r_j (and thus is in Σ_I^k) and each w_j' is in Σ_O^k.

Now, consider an arbitrary $i \geq 1$. Let q_i be the unique state of $\mathfrak{A}$ that is reached from q_{i-1} by processing $\binom{w_{i-1}}{w_{i-1}'}$, and let c_i be the maximal color on the induced run infix. We have $(q_i, c_i) \in r_{i-1}(q_{i-1}, c_{i-1})$, i.e., (q_i, c_i) is a legal move for Player O in $\mathcal{G}_k$ to extend the play prefix $r_0(q_0, c_0) \cdots (q_{i-1}, c_{i-1})r_i$. Accordingly, we define $\sigma'(r_0(q_0, c_0) \cdots (q_{i-1}, c_{i-1})r_i) = (q_i, c_i)$. Player O reacts by picking some $r_{i+1} \in \mathfrak{R}$, which has some witness w_{i+1}. In $\Gamma_k(L(\mathfrak{A}))$ we let Player I pick the letters of w_{i+1} during the next k rounds, which yield k letters $w_i' \in \Sigma_O^k$ determined by σ. Thus, we are in the situation above for $i + 1$, i.e., we have concluded the definition of σ'.

It remains to show that σ' is indeed winning. Fix a play $r_0(q_0, c_0)r_1(q_1, c_1)r_2(q_2, c_2)\cdots$ that is consistent with σ', and let $\binom{w_0 w_1 w_2 \cdots}{w_0' w_1' w_2' \cdots}$ be the play in $\Gamma_k(L(\mathfrak{A}))$ constructed during the simulation. By construction, each w_i is a witness of r_i. An induction shows that q_{i+1} is the unique state of $\mathfrak{A}$ reached when processing $\binom{w_i}{w_i'}$ when starting at q_i, and that c_{i+1} is the maximal color encountered on this run infix. As the

unique run of $\mathfrak{A}$ on $\binom{w_0 w_1 w_2 \cdots}{w'_0 w'_1 w'_2 \cdots}$ is accepting (as it is, by construction, an outcome consistent with the winning strategy σ), we conclude that $\limsup_{i\to\infty} c_i$ is even. Hence, the play $r_0(q_0, c_0)r_1(q_1, c_1)r_2(q_2, c_2)\cdots$ is winning for Player O. As the play was chosen arbitrarily, σ' is indeed a winning strategy for Player O in $\mathcal{G}_k$. □

Lemma 2. *If Player O wins $\mathcal{G}_k$, then she wins $\Gamma_{2k-1}(L(\mathfrak{A}))$.*

Proof. Let σ' be a winning strategy for Player O in $\mathcal{G}_k$. We construct a winning strategy σ for Player O in $\Gamma_{2k-1}(L(\mathfrak{A}))$, which will simulate σ'.

So, let Player I pick letters $a_0 \cdots a_{2k-1}$ in round 0 and define $w_0 = a_0 \cdots a_{k-1}$ and $w_1 = a_k \cdots a_{2k-1}$. Furthermore, let $(q_0, c_0) = (q_\iota, \Omega(q_\iota))$, $r_0 = r_{w_0}^{\{(q_0,c_0)\}}$, and $r_1 = r_{w_1}^{r_0(\{(q_0,c_0)\})}$. Then, $r_0(q_0, c_0)r_1$ is a play prefix in $\mathcal{G}_k$ that is consistent with σ'. Then, we are in the following situation for $i = 1$:

- In $\Gamma_{2k-1}(L(\mathfrak{A}))$, Player I has picked $w_0 \cdots w_i$ and Player O has picked $w'_0 \cdots w_{i-2}$ (which is empty for $i = 1$), and
- in $\mathcal{G}_k$, we have a play prefix $r_0(q_0, c_0) \cdots (q_{i-1}, c_{i-1})r_i$ that is consistent with σ' and where each w_j is a witness for r_j. Note that being a play prefix implies $(q_j, c_j) \in \mathrm{dom}(r_j) = r_{j-1}(q_{j-1})$.

Now, pick some arbitrary $i \geq 1$ and consider $(q_i, c_i) = \sigma'(r_0(q_0, c_0) \cdots (q_{i-1}, c_{i-1})r_i)$. As σ' is a strategy for $\mathcal{G}_k$, we have again $(q_i, c_i) \in \mathrm{dom}(r_i) = r_{i-1}(q_{i-1})$. Furthermore, as w_{i-1} is a witness for r_{i-1}, there is some $w'_{i-1} \in \Sigma_O^k$ such that q_i is the unique state $\mathfrak{A}$ reaches when processing $\binom{w_{i-1}}{w'_{i-1}}$ from q_{i-1}, and c_i is the maximal color occurring in this run infix.

Now, we define σ such that it picks the k letters of w'_{i-1} during the next k rounds (independently of the choices of Player I). During these rounds, Player I again determines some $w_{i+1} \in \Sigma_I^k$, inducing $r_{i+1} = r_{w_{i+1}}^{r_i(q_i,c_i)}$. Then, we are in the situation above for $i+1$, i.e., we have concluded the definition of σ.

It remains to show that σ is winning. To this end, fix an outcome $\binom{w_0 w_1 w_2 \cdots}{w'_0 w'_1 w'_2 \cdots}$ that is consistent with σ, where each w_i is in Σ_I^k and each w'_i is in Σ_O^k. Further, let $r_0(q_0, c_0)r_1(q_1, c_1)r_2(q_2, c_2)\cdots$ the play of $\mathcal{G}_k$ constructed during the simulation. By construction, each w_i is a witness of r_i.

As $r_0(q_0, c_0)r_1(q_1, c_1)r_2(q_2, c_2)\cdots$ is consistent with σ' by construction, it is winning for Player O, i.e., $\limsup_{i\to\infty} c_i$ is even. Now, an induction shows that q_{i+1} is the unique state reached by $\mathfrak{A}$ when processing $\binom{w_i}{w'_i}$ starting in q_i, and c_{i+1} is the maximal color on this run infix. From these two properties, we conclude that the run of $\mathfrak{A}$ on $\binom{w_0 w_1 w_2 \cdots}{w'_0 w'_1 w'_2 \cdots}$ is accepting, i.e., the outcome is winning for Player O. As the outcome was chosen arbitrarily, σ is indeed a winning strategy for Player O in $\Gamma_{2k-1}(L(\mathfrak{A}))$. □

3.3 Running Time

Lemma 3. *Given $\mathfrak{A}$ and $k \leq 2^{n^2 c+1}$, one can construct $\mathcal{G}_k$ and determine its winner in exponential time in n, where n and c are the number of states and colors of $\mathfrak{A}$.*

Proof. We argue that $\mathcal{G}_k$ can be expressed as an arena-based parity game (see, e.g., [3] for a definition) of exponential size in n with the same colors as $\mathfrak{A}$. Such a game can be solved in exponential time in n [1]. Thus, it remains to argue that one can construct the parity game in exponential time.

First, we argue that for each partial function $r\colon Q \times C \to 2^{Q\times C}$ one can construct a deterministic finite automaton recognizing the set of witnesses of r. The construction is based on a powerset construction (mirroring the definition of $\delta_{\mathcal{T}}$ and $\delta_{\mathcal{P}}$) and a counter checking that only inputs of length k are accepted. As there are only exponentially many such functions, one can effectively determine $\mathfrak{R}$, i.e., the set of functions whose associated automaton has a non-empty language, in exponential time.

Now, it is straightforward to construct a parity game $(V_I, V_O, E, v_\iota, \Omega')$ in a graph $(V_I \cup V_O, E)$ of exponential size with the following components:

- $V_I = \{v_\iota\} \cup \mathfrak{R} \times (Q \times C)$: vertices of Player I, where v_ι is a fresh initial vertex.
- $V_O = \mathfrak{R}$: vertices of Player O.
- E is the union of the following sets of edges:
  - $\{(v_\iota, r) \mid \mathrm{dom}(r) = \{q_\iota, \Omega(q_\iota)\}\}$: initial moves of Player I, allowing him to pick some $r \in \mathfrak{R}$ with $\mathrm{dom}(r) = \{q_\iota, \Omega(q_\iota)\}$.

- $\{((r,(q,c)),r') \mid \mathrm{dom}(r') = r(q,c)\}$: non-initial moves of Player I allowing him to pick some r' satisfying $\mathrm{dom}(r') = r(q,c)$, where r and (q,c) were previsouly picked by the players.
- $\{(r,(r,(q,c))) \mid (q,c) \in \mathrm{dom}(r)\}$: moves of Player O allowing here to pick some $(q,c) \in \mathrm{dom}(r)$, where r was previously picked by Player I.

- $\Omega'(v) = \begin{cases} c & \text{if } v = (r,(q,c)) \in V_I, \\ \min C & \text{otherwise.} \end{cases}$. Note that the color $\min C$ is neutral in the following sense: Whether a play is winning or not only depends on the colors of the vertices in $V_I \setminus \{v_\iota\}$, but not on vertices in $V_O \cup \{v_\iota\}$.

The resulting parity game implements exactly the rules of the abstract game $\mathcal{G}_k$ and is therefore won by Player O if and only if she wins $\mathcal{G}_k$. □

4 Conclusion

We have presented an exponential-time algorithm approximating the minimal lookahead necessary to win a delay game. Here, we only considered the case of ω-regular winning conditions given by deterministic parity automata.

In the literature, several other types of winning conditions have been considered, e.g., quantitative parity [9] and (quantitative) Linear Temporal Logic [7]. For these types, one can also exhibit an approximation algorithm for the minimal lookahead that has the same complexity as an algorithm deciding the existence of some lookahead using techniques very similar to those introduced here.

Unfortunately, the complexity of the exact optimization problem for games with winning conditions given by deterministic parity automata remains open. Let us conclude by mentioning another open problem on delay games: There is an exponential gap between the upper and lower bounds on the necessary lookahead in delay games with winning conditions given by deterministic Muller automata. The same is true for deciding whether Player O wins the game for some lookahead.

References

[1] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *STOC 2017*, pages 252–263. ACM, 2017.

[2] David Gale and Frank M. Stewart. Infinite games with perfect information. *Annals of Mathematics*, 28:245–266, 1953.

[3] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[4] Michael Holtmann, Lukasz Kaiser, and Wolfgang Thomas. Degrees of lookahead in regular infinite games. *Log. Methods Comput. Sci.*, 8(3), 2012.

[5] Frederick A. Hosch and Lawrence H. Landweber. Finite delay solutions for sequential conditions. In Maurice Nivat, editor, *ICALP 1972*, pages 45–60. North-Holland, Amsterdam, 1972.

[6] Felix Klein and Martin Zimmermann. How much lookahead is needed to win infinite games? *Log. Methods Comput. Sci.*, 12(3), 2016.

[7] Felix Klein and Martin Zimmermann. Prompt delay. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *FSTTCS 2016*, volume 65 of *LIPIcs*, pages 43:1–43:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[8] Sarah Winter and Martin Zimmermann. Finite-state strategies in delay games. *Inf. Comput.*, 272:104500, 2020.

[9] Martin Zimmermann. Games with costs and delays. In *LICS 2017*, pages 1–12. IEEE Computer Society, 2017.