On the Succinctness of Alternating Parity Good-For-Games Automata

Udi Boker
Interdisciplinary Center (IDC) Herzliya, Israel
udiboker@gmail.com

Denis Kuperberg
CNRS, LIP, École Normale Supérieure, Lyon, France
denis.kuperberg@ens-lyon.fr

Karoliina Lehtinen
University of Liverpool, UK
k.lehtinen@liverpool.ac.uk

Michał Skrzypczak
Institute of Informatics, University of Warsaw, Poland
mskrzypczak@mimuw.edu.pl

─── **Abstract** ───

We study alternating parity good-for-games (GFG) automata, i.e., alternating parity automata where both conjunctive and disjunctive choices can be resolved in an online manner, without knowledge of the suffix of the input word still to be read.

We show that they can be exponentially more succinct than both their nondeterministic and universal counterparts. Furthermore, we present a single exponential determinisation procedure and an Exptime upper bound to the problem of recognising whether an alternating automaton is GFG.

We also study the complexity of deciding "half-GFGness", a property specific to alternating automata that only requires nondeterministic choices to be resolved in an online manner. We show that this problem is PSpace-hard already for alternating automata on finite words.

1 Introduction

Good-for-games (GFG) automata were first introduced in [12] as a tool for solving the synthesis problem. The equivalent notion of *history-determinism* was introduced independently in [8] in the context of regular cost functions. Intuitively, a nondeterministic automaton is GFG if nondeterminism can be resolved on the fly, only with knowledge of the input word read so far. GFG automata can be seen as an intermediate formalism between deterministic and nondeterministic ones, with advantages from both worlds. Indeed, like deterministic automata, GFG automata enjoy good compositional properties – useful for solving games and composing automata and trees – and easy inclusion checks [3]. Like nondeterministic automata, they can be exponentially more succinct than deterministic automata [16].

In recent years, much effort has gone into understanding various properties of nondeterministic GFG automata, for instance their relationship with deterministic automata [3, 16, 5, 15], applications in probabilistic model checking [14] and synthesis of LTL, μ-calculus and context-free properties [13, 18], decision procedures for GFGness [19, 16, 2], minimisation [1], and links with recent advances in parity games [10].

Alternating GFG automata are a natural generalisation of nondeterministic GFG automata that enjoy the same compositional properties as nondeterministic GFG automata, while providing more flexibility. As we show in the present work, for some languages alternating GFG parity automata can also be exponentially more succinct, allowing for better synthesis procedures. Indeed, two-player games with winning conditions given by alternating GFG automata are solvable in quasipolynomial time, via a linear reduction to parity games, while for winning conditions given by arbitrary alternating automata, solving games requires determinisation and has therefore double-exponential complexity.

Alternating GFG automata were introduced independently by Colcombet [9] and Quirl [21] while a form of alternating GFG automata with requirements specific to counters were also considered in [17], as a tool to study cost functions on infinite trees. Boker and Lehtinen studied the expressiveness and succinctness of alternating GFG automata in [6], showing that they

- are not more succinct than DFAs on finite words,
- are as expressive as deterministic ones of the same acceptance condition on infinite words,
- and can be determinised with a $2^{\theta(n)}$ size blowup for the Büchi and coBüchi conditions.

Many questions about GFG alternating automata were left open, in particular whether there exists a doubly exponential succinctness gap between alternating GFG and deterministic automata, and the complexity of deciding whether an alternating parity automaton is GFG.

Succinctness of alternating GFG automata. We show that there is a single exponential gap between alternating parity GFG automata and deterministic ones, thereby answering a question left open in [6]. This is in contrast to general alternating automata, for which determinisation incurs a double-exponential size increase. However, we also show that alternating GFG automata can present exponential succinctness compared to both nondeterministic and universal GFG automata. This means that alternating GFG automata can be used to reduce the complexity of solving some games with complex acceptance conditions.

Recognising GFG automata. We give an EXPTIME upper bound to the problem of deciding whether an alternating parity automaton is GFG, matching the known upper bound for recognising nondeterministic parity GFG automata.

We also study the complexity of deciding "half-GFGness", i.e., whether the nondeterminism (or universality) of an automaton is GFG. This property guarantees that composition with games preserves the winner for one of the players. We show that already on finite words, this problem is PSPACE-hard, and it is in EXPTIME for alternating Büchi automata. This shows that a PTIME algorithm for deciding GFGness must exploit the subtle interplay between nondeterminism and universality, and cannot be reduced to checking independently whether each of them is GFG.

Roadmap

We begin with some definitions, after which, in Section 3, we define alternating GFG automata, study their succinctness and the complexity of deciding half-GFGness, that is, whether the nondeterminism within an alternating automaton is GFG. Section 4 provides a

single-exponential determinisation procedure for alternating GFG parity automata. Section 5 shows that GFGness of alternating parity automata is in EXPTIME, using the determinisation of the previous section. Throughout the paper, we provide high-level proof sketches, with detailed technical developments in the full version [4].

2   Preliminaries

Words and automata.   An *alphabet* Σ is a finite nonempty set of letters. A finite (resp. infinite) *word* $u = u_0 \ldots u_k \in \Sigma^*$ (resp. $w = w_0 w_1 \ldots \in \Sigma^\omega$) is a finite (resp. infinite) sequence of letters from Σ. A *language* is a set of words, and the empty word is written ϵ. We denote a set $\{i, i+1, \ldots, j\}$ of integers by $[i, j]$.

An *alternating word automaton* is a tuple $\mathcal{A} = (\Sigma, Q, \iota, \delta, \alpha)$, where: Σ is an alphabet; Q is a finite nonempty set of states; $\iota \in Q$ is an initial state; $\delta \colon Q \times \Sigma \to \mathsf{B}^+(Q)$ is a transition function where $\mathsf{B}^+(Q)$ is the set of positive Boolean formulas (*transition conditions*) over Q; and α, on which we elaborate below, is either an acceptance condition or a transition labelling on top of which an acceptance condition is defined. For a state $q \in Q$, we denote by $\mathcal{A}^q$ the automaton that is derived from $\mathcal{A}$ by setting its initial state ι to q.

An automaton $\mathcal{A}$ is nondeterministic (resp. universal) if all its transition conditions are disjunctions (resp. conjunctions), and it is deterministic if all its transition conditions are just states. We represent the transition function of nondeterministic and universal automata as $\delta \colon Q \times \Sigma \to 2^Q$, and of a deterministic automaton as $\delta \colon Q \times \Sigma \to Q$. A *transition* of an automaton is a triple $(q, a, q') \in Q \times \Sigma \times Q$, sometimes also written $q \xrightarrow{a} q'$.

We denote by $\widehat{\delta} \subseteq \mathsf{B}^+(Q)$ the set of all subformulas of formulas in the image of δ, i.e., all the Boolean formulas that "appear" somewhere in the transition function of $\mathcal{A}$.

Acceptance conditions.   There are various acceptance (winning) conditions, defined with respect to the set of transitions[1] that a path of $\mathcal{A}$ visits infinitely often. (Notice that a transition condition allows for many possible transitions.) We later formally define acceptance of a word w by $\mathcal{A}$ in terms of games, and consider a path of $\mathcal{A}$ on a word w as a play in that game. For nondeterministic automata, a "run" coincides with a "path".

Some of the acceptance conditions are defined on top of a labelling of the transitions rather than directly on the transitions. In particular, in the parity condition, we have $\alpha \colon Q \times \Sigma \times Q \to \Gamma$, where $\Gamma \subseteq \mathbb{N}$ is a finite set of priorities and a path is accepting if and only if the highest priority seen infinitely often on it is even.

The Büchi and coBüchi conditions are special cases of the parity condition with $\Gamma = \{1, 2\}$ and $\Gamma = \{0, 1\}$, respectively. When speaking of Büchi and coBüchi automata, we often refer to α as the set of "accepting transitions", namely the transitions that are mapped to 2 in the Büchi case and to 0 in the coBüchi case. The weak condition is a special case of both the Büchi and coBüchi conditions, in which every path eventually remains in the same priority.

The Rabin and Streett conditions are more involved, yet defined directly on the set T of transitions. A Rabin condition is a set $\{(B_1, G_1), (B_2, G_2), \ldots, (B_k, G_k)\}$, with $B_i, G_i \subseteq T$, and a path ρ is accepting iff for some $i \in [1, k]$, we have that the set $\inf(\rho)$ of transitions that are visited infinitely often in ρ satisfies ($\inf(\rho) \cap B_i = \emptyset$ and $\inf(\rho) \cap G_i \neq \emptyset$). A Streett condition is dual: a set $\{(B_1, G_1), (B_2, G_2), \ldots, (B_k, G_k)\}$, with $B_i, G_i \subseteq Q$, whereby a path ρ is accepting iff for all $i \in [1, k]$, we have ($\inf(\rho) \cap B_i = \emptyset$ or $\inf(\rho) \cap G_i \neq \emptyset$).

---

[1] Acceptance is defined in the literature with respect to either states or transitions; for technical reasons we prefer to work with acceptance on transitions.

Sizes and types of automata.   The size of $\mathcal{A}$ is the maximum of the alphabet size, the number of states, the transition function length, which is the sum of the transition condition lengths over all states and letters, and the acceptance condition's index, which is 1 for weak, Büchi and coBüchi, $|\Gamma|$ for parity, and k for Rabin and Street.

We sometimes abbreviate automata types by three-letter acronyms in $\{D, N, U, A\} \times \{F, W, B, C, P, R, S\} \times \{A, W\}$. The first letter stands for the transition mode, the second for the acceptance condition, and the third indicates that the automaton runs on finite or infinite words. For example, DPW stands for a deterministic parity automaton on infinite words.

Games and strategies.   Some of our technical proofs use standard concepts of an arena, a game, a winning strategy, etc. For the sake of completeness, we provide precise mathematical definitions of these objects in the full version [4]. Here we will just overview the involved concepts.

First, we work with two-player games of perfect information, where the players are Eve and Adam. These games are played on graphs (called arenas). Most of the considered games are of infinite duration and their winning condition is expressed in terms of the infinite sequences of edges taken during the play. We invoke results of determinacy (one of the players has a winning strategy), as well as of *positional determinacy* (one of the players has a strategy that depends only on the last position of the play).
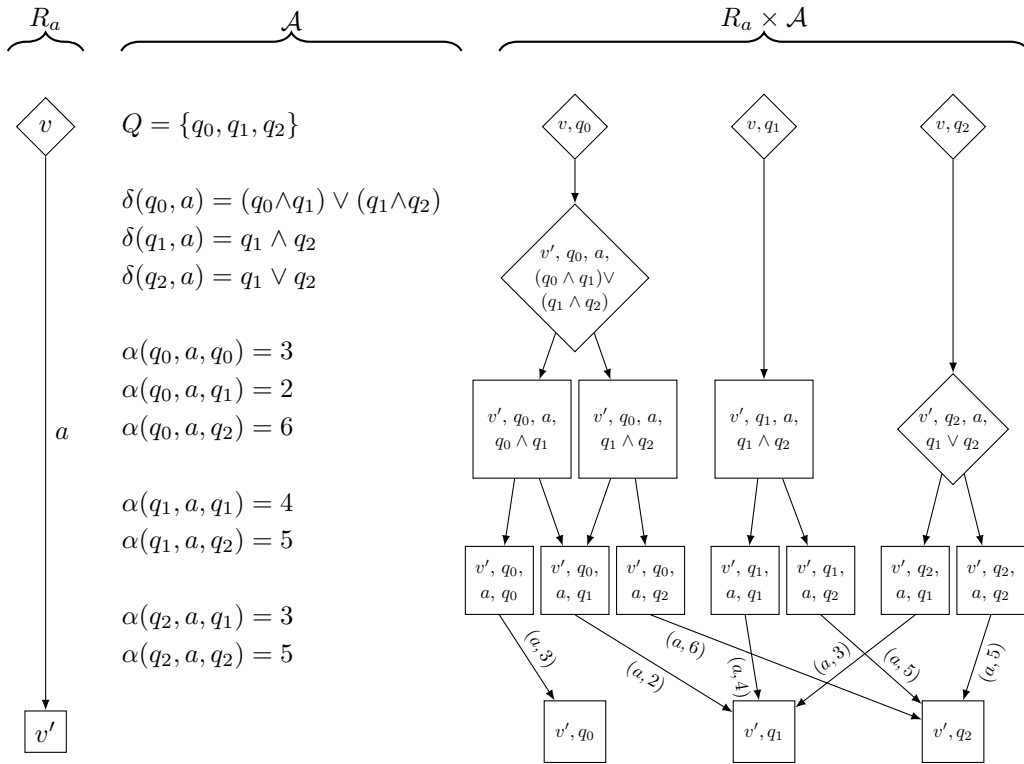
Figure 1 A one-step arena over a letter $a \in \Sigma$, obtained as a product of a simple arena R_a with the alternating parity automaton $\mathcal{A}$. In this example v is controlled by Eve and v' by Adam. The transitions with no label are labelled by ϵ. Diamond-shaped positions belong to Eve and square-shaped positions belong to Adam.
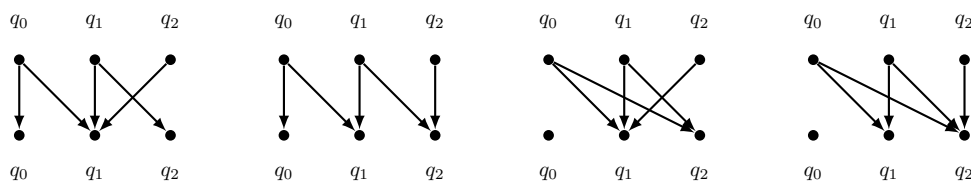
Figure 2 The four possible boxes corresponding to Eve's choices in the one-step arena of Figure 1. (All edges should be labelled with a, which we omit for better readability.)

Model-checking games. To represent the semantics of an alternating automaton $\mathcal{A}$, we treat the Boolean formulas that appear in the transition conditions of $\mathcal{A}$ as games. More precisely, given a letter $a \in \Sigma$ we represent the transition conditions $q \mapsto \delta(q,a) \in \mathsf{B}^+(Q)$ as the *one-step arena* over a (see Figure 1). A play over this arena begins in a state $q \in Q$; then the players go down the formula $\delta(q,a)$ with Eve resolving disjunctions and Adam resolving conjunctions; and finally they reach an atom $q' \in Q$ and the play stops. This means that a play over the one-step arena over a results in a transition of the form $q \xrightarrow{a} q'$.

The language $L(\mathcal{A})$ of an alternating automaton $\mathcal{A}$ over an alphabet Σ is defined via the *model-checking game*, defined for an automaton $\mathcal{A}$ and a word $w = a_0a_1a_2\cdots \in \Sigma^\omega$. A *configuration* of this game is a state q of $\mathcal{A}$ and a position $i \in \omega$ of w, starting at $(\iota, 0)$. In the ith round, starting at configuration (q_i, i), the players play on the one-step arena from q_i over a_i, resulting in a transition $q_i \xrightarrow{a_i} q_{i+1}$. The next configuration is $(q_{i+1}, i+1)$. The acceptance condition of $\mathcal{A}$ becomes the winning condition of this game. $\mathcal{A}$ *accepts* w if Eve has a winning strategy in this game.

For technical convenience, we define (in the full version) the model-checking game in terms of a *synchronised product* of the word w (treated as an infinite graph) and the automaton $\mathcal{A}$. Synchronised products turn out to be useful in the analysis of various games presented in this paper and are used throughout the technical version of the paper [4].

▶ **Definition 1.** *Given an alternating automaton $\mathcal{A}$, we denote by $\overline{\mathcal{A}}$ the dual automaton: it has the same alphabet, set of states, and initial state. Its transition conditions $\delta_{\overline{\mathcal{A}}}(q,a)$ are obtained from those of $\mathcal{A}$ by replacing each disjunction $\vee$ with conjunction $\wedge$ and vice versa. Its acceptance condition is the dual of $\mathcal{A}$'s condition. (In parity automata, all priorities are increased by 1.) $\overline{\mathcal{A}}$ recognises the complement $L(\mathcal{A})^c$ of $L(\mathcal{A})$.*

Boxes. Another technical concept that we use is that of *boxes* (see Figure 2), which describe Eve's *local* strategies for resolving disjunctions within a transition condition. Consider an alternating automaton $\mathcal{A}$ and a letter $a \in \Sigma$. Moreover, fix a strategy σ of Eve that resolves disjunctions in all the transition conditions $\delta(q,a)$ for $q \in Q$. Now, the *box* of $\mathcal{A}$, a, and σ is a subset of $Q \times \Sigma \times Q$ and contains a triple (q,a,q') iff σ resolves disjunctions of $\delta(q,a)$ in such a way that Adam (resolving conjunctions) can reach the atom q'. In other words, this box contains (q,a,q') if there is a play consistent with σ on $\delta(q,a)$ that reaches the atom q'. We use β to denote single boxes and by $\mathsf{B}_{\mathcal{A},a}$ we denote the set of all boxes of $\mathcal{A}$ and a, while $\mathsf{B}_{\mathcal{A}}$ denotes the union $\bigcup_{a \in \Sigma} \mathsf{B}_{\mathcal{A},a}$. We give a more formal definition based on synchronised products in the full version [4].

▶ **Definition 2.** *Given a sequence of boxes $\pi = b_0, b_1, \ldots$ of an automaton $\mathcal{A}$ and a path $\rho = (q_0, a_0, q_1), (q_1, a_1, q_2), \ldots$, we say that ρ is a path of π if for every i we have $(q_i, a_i, q_{i+1}) \in b_i$. The sequence π is said to be universally accepting if every path in π is accepting in $\mathcal{A}$.*
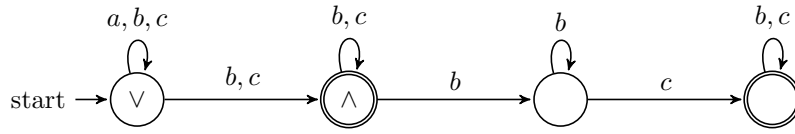
Figure 3 Alternating weak automaton accepting words over $\{a, b, c\}$ in which a occurs finitely often and c occurs infinitely often. Omitted transitions lead to a rejecting sink.

Intuitively, a sequence of boxes π as above represents a particular positional strategy σ of Eve in the model-checking game over the word $w = a_0 a_1 a_2 \ldots$ In that case, a path of π corresponds to a possible play of this game consistent with σ, and the sequence is universally accepting if and only if the strategy is winning.

3   Good-For-Games Alternating Automata

Good-for-games (GFG) nondeterministic automata are automata in which the nondeterministic choices can be resolved without looking at the future of the word. For example, consider an automaton that consists of a nondeterministic choice between a component that accepts words in which a occurs infinitely often and a component that accepts words in which a occurs finitely often. This automaton accepts all words but is not GFG since the nondeterministic choice of component cannot be resolved without knowing the whole word.

To extend this definition to alternating automata, we must look both at its nondeterminism and universality and require that both can be resolved without knowledge of the future. The following letter games capture this intuition.

▶ **Definition 3** (Letter games [6]). *Given an alternating automaton $\mathcal{A}$, Eve's letter game proceeds at each turn from a state q of $\mathcal{A}$, starting from the initial state of $\mathcal{A}$, as follows:*
- *Adam chooses a letter a,*
- *Adam and Eve play on the one-step arena over a from q to a new state q', where Eve resolves disjunctions and Adam conjunctions.*

A play of the letter game thus generates a word w and a path ρ of $\mathcal{A}$ on w. Eve wins this play if either $w \notin L(\mathcal{A})$ or ρ is accepting in $\mathcal{A}$.

Adam's letter game is similar, except that Eve chooses letters and Adam wins if either $w \in L(\mathcal{A})$ or the path ρ is rejecting.

▶ **Definition 4** (GFG automata [6]). *An automaton $\mathcal{A}$ is $\exists$-GFG if Eve wins her letter game; it is $\forall$-GFG if Adam wins his letter game. Finally, $\mathcal{A}$ is GFG if it is both $\exists$-GFG and $\forall$-GFG.*

As shown in [6, Theorem 8], an automaton $\mathcal{A}$ is GFG if and only if it is indeed "good for playing games", in the sense that its product with every game whose winning condition is $L(\mathcal{A})$ preserves the winner of the game.

▶ **Example 5.** The automaton in Figure 3 accepts the language L of words in which a occurs finitely often and c occurs infinitely often. Here Eve loses her letter game: Adam can play c until Eve takes the transition to the second state, and then play a followed by c^ω. Conversely, Eve wins Adam's letter game: her strategy is to play b, take the transition to the second state an keep playing b until Adam takes the transition into the third state, after which she plays c once and then b^ω. This automaton is neither $\exists$-GFG nor $\forall$-GFG, and taking its product with games with L as winning condition does not preserve the winner of the game.
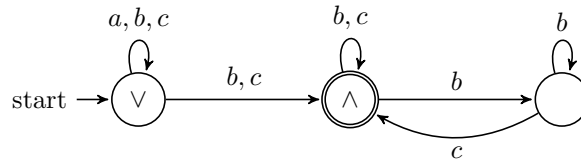
Figure 4 Alternating coBüchi ∀-GFG automaton accepting words over $\{a, b, c\}$ in which a occurs finitely often and c occurs infinitely often.

In contrast, the automaton in Figure 4 is ∀-GFG but not ∃-GFG. Indeed, Adam's winning strategy in his letter game is to resolve the conjunction from the middle state by always moving to the right-hand state when Eve plays b. This forces Eve to choose between playing c infinitely many times (in which case, the word is in the language) or letting Adam build a rejecting run. Taking its product with *one-player games* with winning condition L preserves the winner whenever Eve is the player controlling all positions. However, this is not the case for one-player games where Adam is the sole player.

3.1 Alternating GFG vs. Nondeterministic and Universal Ones

We show in this section that alternating GFG automata can be more succinct than both nondeterministic and universal GFG automata.

▶ **Lemma 6.** *There is a family $(\mathcal{C}_n)_{n \in \mathbb{N}}$ of alternating GFG $\{0, 1, 2\}$-parity automata of size linear in n over a fixed alphabet, such that every nondeterministic GFG parity automaton and universal GFG parity automaton for $L(\mathcal{C}_n)$ is of size $2^{\Omega(n)}$.*

Proof. From [16], there is a family $(\mathcal{A}_n)_{n \in \mathbb{N}}$ of GFG-NCWs with n states over a fixed alphabet Σ, such that every DPW for $L_n = L(\mathcal{A}_n)$ is of size $2^{\Omega(n)}$. For every $n \in \mathbb{N}$, let $\mathcal{B}_n$ be the dual of $\mathcal{A}_n$, so $\mathcal{B}_n$ is a UBW accepting $\overline{L_n}$. We build an APW $\mathcal{C}_n$ over Σ of size linear in n, by setting its initial state to move to the initial state of $\mathcal{A}_n$ when reading the letter $a \in \Sigma$ and to the initial state of $\mathcal{B}_n$ when reading the letter $b \in \Sigma$. The acceptance condition of $\mathcal{C}_n$ is a parity condition with priorities $\{0, 1, 2\}$: accepting transitions of $\mathcal{A}_n$ are assigned priority 0, and accepting transitions of $\mathcal{B}_n$ priority 2. Other transitions have priority 1.

The automaton C_n is represented below:
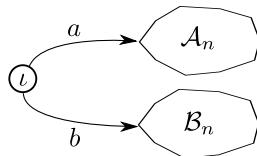
Observe that $L(\mathcal{C}_n) = aL_n \cup b\overline{L_n}$, and that $\mathcal{C}_n$ is GFG: its initial state has only deterministic transitions, and over the $\mathcal{A}_n$ and $\mathcal{B}_n$ components, the strategy to resolve the nondeterminism and universality, respectively, follows the strategy to resolve the nondeterminism of $\mathcal{A}_n$, which is guaranteed due to $\mathcal{A}_n$'s GFGness.

Consider a GFG UPW $\mathcal{E}_n$ for $L(\mathcal{C}_n)$, and let q be a state to which $\mathcal{E}_n$ moves when reading a, according to some strategy that witnesses $\mathcal{E}_n$'s GFGness. Then $\mathcal{E}_n^q$ is a GFG UPW for L_n. Its dual is therefore a GFG NPW $\mathcal{E}_n'$ for $\overline{L_n}$.

Since $\mathcal{A}_n$ is a GFG NPW for L_n, by [3, Theorem 4] we obtain a DPW for L_n of size $|\mathcal{A}_n||\mathcal{E}_n'|$. By choice of L_n, this DPW must be of size $2^{\Omega(n)}$, and since $\mathcal{A}_n$ is of size n, it follows that $\mathcal{E}_n'$, and hence $\mathcal{E}_n$, must be of size $2^{\Omega(n)}$. By a symmetric argument, every GFG NPW for $L(\mathcal{C}_n)$ must also be of size $2^{\Omega(n)}$. ◀

Informally, the language L_n above describes a set of threads, of which at least one eventually satisfies a safety property. Then, the above construction can be understood as describing a property of reactive systems where, depending on the input, the system guarantees either that there is a thread that eventually satisfies a safety property, or that all threads satisfy a liveness (Büchi) property. The GFG alternating automaton can then be used to solve in polynomial time games with such languages as winning condition, for example in the context of synthesis: the product of the game arena and the alternating automaton for L_n is a parity game with 3 priorities with the same winner as the original game. In contrast, a DPW, GFG NPW and GFG UPW for the same language would all be exponentially larger.

3.2　Deciding Half-GFGness

In order to decide GFGness, it is enough to be able to decide the $\exists$-GFG property on the automaton and its dual. A natural first approach is therefore to study the complexity of deciding whether an APW is $\exists$-GFG. Yet, we will show that already on finite words, this problem is PSpace-hard, while we conjecture that deciding GFGness is in PTime.

▶ **Lemma 7.** *Deciding whether an AFA is $\exists$-GFG is PSpace-hard.*

Proof. We reduce from NFA universality: starting from an NFA $\mathcal{A}$, we build an AFA $\mathcal{B}$ based on the dual of $\mathcal{A}$, with an additional non-GFG choice to be resolved by Eve. This AFA $\mathcal{B}$ is $\exists$-GFG if and only if $L(\mathcal{B}) = \emptyset$, which happens if and only if $L(\mathcal{A}) = \Sigma^*$. We crucially use the fact that $\mathcal{B}$ is not necessarily $\forall$-GFG.

Let $\mathcal{A}$ be an NFA over an alphabet $\Sigma = \{a, b\}$ and $\bar{\mathcal{A}}$ its dual. We want to check whether $L(\mathcal{A}) = \Sigma^*$. We build an AFA $\mathcal{B}$, as depicted below, by first making Eve guess the second letter. If her guess is wrong, the automaton proceeds to a rejecting sink state $\bot$. Otherwise, it proceeds to the initial state of $\bar{\mathcal{A}}$. The size of $\mathcal{B}$ is linear in the size of $\mathcal{A}$.
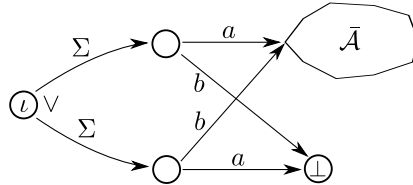
If $L(\bar{\mathcal{A}}) = \emptyset$, then $L(\mathcal{B}) = \emptyset$, so $\mathcal{B}$ is trivially $\exists$-GFG. However, if there is some $u \in L(\bar{\mathcal{A}})$, then Adam has a winning strategy in Eve's letter game on $\mathcal{B}$. This strategy consists of playing a, then playing the letter that brings Eve to $\bot$, and finally playing u. The resulting word is in $L(\mathcal{B}) = \Sigma^2 L(\mathcal{A})$, so this witnesses that $\mathcal{B}$ is not $\exists$-GFG. We obtain that $L(\mathcal{A}) = \Sigma^* \Leftrightarrow L(\bar{\mathcal{A}}) = \emptyset \Leftrightarrow \mathcal{B}$ is $\exists$-GFG, which is the wanted reduction. ◀

For Büchi automata, and so in particular for finite words, we can give an ExpTime algorithm for this problem.

▶ **Lemma 8.** *Deciding whether an ABW is $\exists$-GFG is in ExpTime.*

Proof. It is shown in [6, Lemma 23] that removing alternation from an ABW $\mathcal{A}$ using the breakpoint construction [20] yields an NBW $\mathcal{B}$ such that if $\mathcal{A}$ is $\exists$-GFG then $\mathcal{B}$ is GFG. Moreover, the converse also holds: if $\mathcal{B}$ is GFG then $\mathcal{A}$ is $\exists$-GFG, since playing Eve's letter game in $\mathcal{B}$ is more difficult for Eve than playing it in $\mathcal{A}$. This means that starting from an ABW $\mathcal{A}$, we can build an exponential size NBW $\mathcal{B}$ via breakpoint construction, and test whether $\mathcal{B}$ is GFG via the algorithm from [2], in time polynomial with respect to $\mathcal{B}$. Overall, this yields an ExpTime algorithm deciding whether $\mathcal{A}$ is $\exists$-GFG. ◀

4 Determinisation of Alternating GFG Parity Automata

In this section we provide a procedure that, given an alternating GFG parity automaton, produces an equivalent deterministic parity automaton with single-exponentially many states. To do so, we first provide an alternation-removal procedure that preserves GFG status. Then, we apply this procedure to both the input automaton and its complement and use the GFG strategies in these two automata to determinise the input automaton. Our proofs, in [4] rely on some analysis of when GFG strategies can use the history of the word, rather than the full history of the play (which also includes the choices of how to resolve the nondeterminism and universality), and on the memoryless determinacy of parity games.

Our method for going from alternating to nondeterministic automata is similar to that of Dax and Klaedtke [11]: they take a nondeterministic automaton that recognises the universally-accepting words in $(B_A)^\omega$ and add nondeterminism that upon reading a letter $a \in \Sigma$ chooses a box in B_A over a. Yet in our approach, in order to guarantee that the outcome preserves GFGness, the intermediate automaton is deterministic.

4.1 Alternation Removal in GFG Parity Automata

▶ **Theorem 9.** *Consider an alternating parity automaton $\mathcal{A}$ with n states and index k. There exists a nondeterministic parity automaton $\mathtt{box}(\mathcal{A})$ with $2^{O(nk \log nk)}$ states that is equivalent to $\mathcal{A}$ such that if $\mathcal{A}$ is GFG then $\mathtt{box}(\mathcal{A})$ is also GFG.*

In Section 5, where we discuss decision procedures, we will show that $\mathtt{box}(\mathcal{A})$ is GFG *exactly* when $\mathcal{A}$ is $\exists$-GFG. For now, the rest of this section is devoted to the proof of Theorem 9, of which a detailed version can be found in the full version [4].

▶ **Lemma 10.** *Consider an alternating parity automaton $\mathcal{A}$ with n states and index k. Then there exists a deterministic parity automaton $\mathcal{B}$ with $2^{O(nk \log nk)}$ states over the alphabet $B_\mathcal{A}$ that recognises the set of universally-accepting words for $\mathcal{A}$. If $\mathcal{A}$ is a Büchi automaton, then $\mathcal{B}$ can also be taken as Büchi, and in general the parity index of the automaton $\mathcal{B}$ is linear in the number of transitions of $\mathcal{A}$.*

Proof sketch. We first construct a nondeterministic parity (resp. coBüchi) automaton over the alphabet $B_\mathcal{A}$ that recognises the complement of the set of universally-accepting words for $\mathcal{A}$. This automaton is easy to build: it guesses a path that is not accepting, and has the dual acceptance condition to $\mathcal{A}$. We then obtain the automaton $\mathcal{B}$ by determinising and complementing this automaton.   ◀

We now build the automaton $\mathtt{box}(\mathcal{A})$ of Theorem 9. It is the same as the automaton $\mathcal{B}$ of Lemma 10, except that the alphabet is Σ and the transition function is defined as follows: For every state p of $\mathcal{B}$ and $a \in \Sigma$, we have $\delta_{\mathtt{box}(\mathcal{A})}(p, a) := \bigcup_{\beta \in B_{(\mathcal{A},a)}} \delta_\mathcal{B}(p, \beta)$.

In other words, the automaton $\mathtt{box}(\mathcal{A})$ reads a letter a, nondeterministically guesses a box $\beta \in B_{\mathcal{A},a}$, and follows the transition of $\mathcal{B}$ over β. Thus, the runs of $\mathtt{box}(\mathcal{A})$ over a word $w = w_0 w_1 w_2 \cdots \in \Sigma^\omega$ are in bijection with sequences of boxes $(\beta_i)_{i \in \mathbb{N}}$ such that $\beta_i \in B_{\mathcal{A},w_i}$ for all $i \in \mathbb{N}$.

Fix an infinite word $w \in \Sigma^\omega$. Our aim is to prove that $w \in L(\mathcal{A}) \Leftrightarrow w \in L(\mathtt{box}(\mathcal{A}))$.

▶ **Lemma 11.** *There exists a bijection between positional strategies of Eve in the acceptance game of $\mathcal{A}$ over w and runs of $\mathtt{box}(\mathcal{A})$ over w. Moreover, a strategy is winning if and only if the corresponding run is accepting. Thus $L(\mathcal{A}) = L(\mathtt{box}(\mathcal{A}))$.*

▶ **Remark 12.** The above alternation-removal procedure also extends to alternating Rabin automata but fails for alternating Streett automata $\mathcal{A}$: since Streett games are not positionally determined for Eve, the acceptance game of $\mathcal{A}$ over a word w is not positionally determined for Eve.

▶ **Lemma 13.** *For an alternating $\exists$-GFG parity automaton $\mathcal{A}$, the automaton $\mathrm{box}(\mathcal{A})$ is GFG.*

Intuitively, this is because the construction of $\mathrm{box}(\mathcal{A})$ preserves the nondeterminism of $\mathcal{A}$.

4.2 Single-Exponential Determinisation

The aim of this section is to prove the following determinisation theorem.

▶ **Theorem 14.** *If $\mathcal{A}$ is an alternating parity GFG automaton then there exists a deterministic parity automaton $\mathcal{D}$ that recognises the same language and has size at most exponential in the size of $\mathcal{A}$. Moreover, the parity index of $\mathcal{D}$ is the same as that of $\mathcal{A}$.*

▶ **Remark 15**. Theorem 9 and [3, Theorem 4], which uses an NRW-GFG and its complement NRW-GFG to obtain a DRW, together give an exponential deterministic parity automaton for $L(\mathcal{A})$. However, the index of $\mathcal{A}$ might not be preserved. On the other hand, from [6, Theorem 19] we know that there exists a deterministic parity automaton equivalent to $\mathcal{A}$ with the same index, but it might have more than exponentially many states. Here we are able to guarantee both the preservation of the index and an exponential upper bound on the size of the deterministic automaton.

Observe that Theorem 9 can be applied both to $\mathcal{A}$ and its dual. Therefore, we can fix a pair of nondeterministic GFG parity automata $\mathrm{box}(\mathcal{A})$ and $\mathrm{box}(\bar{\mathcal{A}})$ that recognise $L(\mathcal{A})$ and $L(\mathcal{A})^c$ respectively and are both of size exponential in $\mathcal{A}$. We use the automata $\mathcal{A}$, $\mathrm{box}(\mathcal{A})$, and $\mathrm{box}(\bar{\mathcal{A}})$ to construct two auxiliary games $G(\mathcal{A})$ and $G'(\mathcal{A})$.

The game $G(\mathcal{A})$ proceeds from a configuration consisting of a pair (p, q) of states from $\mathrm{box}(\bar{\mathcal{A}})$ and $\mathcal{A}$ respectively, starting from their initial states, as follows:
- Adam chooses a letter $a \in \Sigma$;
- Eve chooses a transition $p \xrightarrow{a} p'$ in $\mathrm{box}(\bar{\mathcal{A}})$;
- Eve and Adam play on the one-step arena over a from q to a new state q'.

A play in $G(\mathcal{A})$ consists of a run ρ in $\mathrm{box}(\bar{\mathcal{A}})$ and a path ρ' in $\mathcal{A}$. It is winning for Eve if either ρ is accepting in $\mathrm{box}(\bar{\mathcal{A}})$ (in which case $w \notin L(\mathcal{A})$), or ρ' is accepting in $\mathcal{A}$.

If $\mathcal{A}$ is $\exists$-GFG and $\mathrm{box}(\bar{\mathcal{A}})$ is GFG, Eve has a winning strategy in $G(\mathcal{A})$ consisting of building a run in $\mathrm{box}(\bar{\mathcal{A}})$ using her GFG strategy in $\mathrm{box}(\bar{\mathcal{A}})$ and a path in $\mathcal{A}$ using her $\exists$-GFG strategy in $\mathcal{A}$. This guarantees that if $w \in L(\mathcal{A})$ then the path in $\mathcal{A}$ is accepting, and otherwise the run in $\mathrm{box}(\bar{\mathcal{A}})$ is accepting.

We then argue that as the winning condition of $G(\mathcal{A})$ is a Rabin condition, Eve also has a winning strategy that is positional in $\mathcal{A}$, that is, which only depends on the history of the word and the current position. (Interestingly, the question of whether Eve can resolve the nondeterminism in a class of alternating GFG automata with only the knowledge of the word read so far does not tightly correspond to whether the acceptance condition of this class is memoryless. For example, it does hold for the generalised-Büchi condition, though it is not memoryless.)

▶ **Remark 16**. There is some magic here: both the GFG strategies of Eve in $\mathcal{A}$ and in $\mathrm{box}(\bar{\mathcal{A}})$ may require exponential memory, yet, when she needs to satisfy the disjunction of the two conditions, no more memory is needed. In a sense, the states of $\mathcal{A}$ provide the memory for $\mathrm{box}(\bar{\mathcal{A}})$ and the states of $\mathrm{box}(\bar{\mathcal{A}})$ provide the memory for $\mathcal{A}$.

The game $G'(\mathcal{A})$ is similar, except that Adam is given control of $\mathtt{box}(\mathcal{A})$ and Eve is in charge of letters. This time Adam wins a play, consisting of a run of $\mathtt{box}(\mathcal{A})$ and a path in $\mathcal{A}$, if either the path of $\mathcal{A}$ is rejecting or the run of $\mathtt{box}(\mathcal{A})$ is accepting.

Accordingly, if $\mathcal{A}$ is GFG, then he can win by using the $\exists$-GFG strategy in $\mathtt{box}(\mathcal{A})$ and the $\forall$-GFG strategy in $\mathcal{A}$. Then if $w \in L(\mathcal{A})$, the run in $\mathtt{box}(\mathcal{A})$ is accepting, and otherwise the path of $\mathcal{A}$ is rejecting. As before, he also has a positional winning strategy in $G'(\mathcal{A})$.

We are now ready to build the deterministic automaton from a GFG APW $\mathcal{A}$, using positional winning strategies σ and τ for Eve and Adam in $G(\mathcal{A})$ and $G'(\mathcal{A})$, respectively.

Let $\mathcal{D}$ be the automaton with states of the form (q, p_1, p_2), with q a state of $\mathcal{A}$, p_1 a state of $\mathtt{box}(\mathcal{A})$ and p_2 a state of $\mathtt{box}(\bar{\mathcal{A}})$. A transition of D over a moves to (q', p_1', p_2') such that moving from (q, p_1) to (q', p_1) is consistent with τ; and moving from (q, p_2) to (q', p_2') is consistent with σ. The acceptance condition of $\mathcal{D}$ is inherited from $\mathcal{A}$.

▶ **Lemma 17.** *For a GFG APW $\mathcal{A}$ and $\mathcal{D}$ built as above, $L(\mathcal{A}) = L(\mathcal{D})$.*

▶ Remark 18. To extend this construction to an alternating GFG Rabin automaton $\mathcal{A}$, we would need to remove alternations from both $\mathcal{A}$ and its dual while preserving GFGness. However, the dual is a Streett automaton, for which we cannot invoke positional determinacy.

5   Deciding GFGness of Alternating Automata

We use the development of the last section to show that deciding whether an APW is GFG is in EXPTIME. This matches the best known upper bound for the same problem on NPW.

The main result of this section is the following theorem.

▶ **Theorem 19.** *There exists an EXPTIME algorithm that takes as input an alternating parity automaton $\mathcal{A}$ and decides whether $\mathcal{A}$ is GFG.*

The idea is to construct the (exponential size) NPWs $\mathtt{box}(\mathcal{A})$ and $\mathtt{box}(\bar{\mathcal{A}})$ for $L(\mathcal{A})$ and $L(\mathcal{A})^{\mathsf{c}}$ respectively, which are GFG if and only if $\mathcal{A}$ is $\exists$-GFG and $\forall$-GFG respectively. Then, it remains to check whether both are indeed GFG. Since we don't have a polynomial procedure to check this, instead, we will build a game which Eve wins if and only if *both* are indeed GFG, and which we can solve in exponential time with respect to the size of $\mathcal{A}$.

First, we observe the following reciprocal of Lemma 13.

▶ **Lemma 20.** *If $\mathtt{box}(\mathcal{A})$ is GFG then $\mathcal{A}$ is $\exists$-GFG.*

Proof. Assume that $\mathtt{box}(\mathcal{A})$ is GFG and consider a strategy witnessing this. Such a strategy can be easily turned into a function $\sigma' \colon \Sigma^+ \to \mathsf{B}_{\mathcal{A}}$ that, given a word $w \in L(\mathcal{A})$ produces a universally accepting word of boxes of $\mathcal{A}$. Now, due to the definition of a box, each such box defines a positional strategy of Eve in the respective one-step game. This allows us to construct a winning strategy of Eve in the letter game over $\mathcal{A}$.   ◀

Thus, $\mathcal{A}$ is GFG if and only if both $\mathtt{box}(\mathcal{A})$ and $\mathtt{box}(\bar{\mathcal{A}})$ are GFG. To decide this, we consider a game G'' where Adam plays letters and Eve produces runs of the automata $\mathtt{box}(\mathcal{A})$ and $\mathtt{box}(\bar{\mathcal{A}})$ in parallel. The winning condition of G'' requires that at least one of the constructed runs must be accepting.

Now, each sequence of letters given by Adam belongs either to the language of $\mathtt{box}(\mathcal{A})$ or to $\mathtt{box}(\bar{\mathcal{A}})$ and therefore, a winning strategy of Eve in G'' must comprise of two strategies witnessing GFGness of both $\mathtt{box}(\mathcal{A})$ and $\mathtt{box}(\bar{\mathcal{A}})$. Dually, if both $\mathtt{box}(\mathcal{A})$ and $\mathtt{box}(\bar{\mathcal{A}})$ are GFG then Eve wins G'' by playing the two strategies in parallel.

It remains to show that G'' is solvable in EXPTIME. Its winning condition is a disjunction of parity conditions, with index linear in the number of transitions of $\mathcal{A}$. This winning condition is recognised by a deterministic parity automaton of exponential size with polynomial index. To solve G'', we take its product with this deterministic automaton that recognises its winning condition, and solve the resulting parity game with an algorithm that is polynomial in the size of the game whenever, like here, the number of priorities is logarithmic in the size of the game, for instance [7]. Details of this construction and its complexity are in the full version [4].

6   Conclusions

The results obtained in this work shed new light on where alternating GFG automata resemble nondeterministic ones, and where they differ. Overall, our results show that allowing GFG alternations add succinctness without significantly increasing the complexity of determinisation nor decision procedures.

In particular, we show that alternating parity GFG automata can be exponentially more succinct than any equivalent nondeterministic GFG automata, yet this succinctness does not become double exponential when compared to deterministic automata, answering a question from [6]. Some further succinctness problems are left open here, such as the possibility of a doubly exponential gap between alternating GFG automata of stronger acceptance conditions and deterministic ones, as well as between ∃-GFG parity automata and deterministic ones.

We also show that the interplay between the two players can be used to decide whether an automaton is GFG without deciding ∃-GFG and ∀-GFG separately, yielding an EXPTIME algorithm. This matches the current algorithms for deciding GFGness on non-deterministic automata. Bagnol and Kuperberg conjectured that GFGness is PTIME decidable for non-deterministic parity automata of fixed index [2]; we extend this conjecture to alternating automata.

It then becomes interesting to ask how to build an alternating automaton GFG. Indeed, Henzinger and Piterman [12] proposed a transformation of nondeterministic automata into GFG automata, which, despite in some cases leading to a deterministic automaton, is, conceptually, a much simpler procedure than determinisation. Indeed, in many examples of non-GFG automata, adding transitions suffices to obtain a GFG one. We leave finding such a procedure for alternating automata as future work.

References

1   Bader Abu Radi and Orna Kupferman. Minimizing GFG transition-based automata. In *Proceedings of ICALP*, pages 100:1–100:16, 2019.

2   Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proceedings of FSTTCS*, pages 16:1–16:14, 2018.

3   Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proceedings of ICALP*, pages 89–100, 2013.

4   Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michał Skrzypczak. On the succinctness of alternating parity good-for-games automata, 2020. `arXiv:2009.14437`.

5   Udi Boker, Orna Kupferman, and Michał Skrzypczak. How deterministic are good-for-games automata? In *Proceedings of FSTTCS*, pages 18:1–18:14, 2017.

6   Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In *Proceedings of CONCUR*, 2019.

7   Cristian S Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of STOC*, pages 252–263, 2017.

8   Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proceedings of ICALP*, pages 139–150, 2009.

9   Thomas Colcombet. Fonctions régulières de coût. Habilitation thesis, 2013.

10   Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata: New tools for infinite duration games. In *Proceedings of FOSSACS*, pages 1–26, 2019.

11   Christian Dax and Felix Klaedtke. Alternation elimination by complementation. In *Proceedings of LPAR*, pages 214–229, 2008.

12   Thomas Henzinger and Nir Piterman. Solving games without determinization. In *Proceedings of CSL*, pages 395–410, 2006.

13   Simon Iosti and Denis Kuperberg. Eventually safe languages. In *Proceedings of DLT*, pages 192–205, 2019.

14   Joachim Klein, David Müller, Christel Baier, and Sascha Klüppelholz. Are good-for-games automata good for probabilistic model checking? In *Proceedings of LATA*, pages 453–465, 2014.

15   Denis Kuperberg and Anirban Majumdar. Computing the width of non-deterministic automata. *Logical Methods in Computer Science*, 15(4), 2019.

16   Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In *Proceedings of ICALP*, pages 299–310, 2015.

17   Denis Kuperberg and Michael Vanden Boom. Quasi-weak cost automata: A new variant of weakness. In *Proceedings of FSTTCS*, pages 66–77, 2011.

18   Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 689–702, 2020.

19   Christof Löding and Stefan Repke. Decidability Results on the Existence of Lookahead Delegators for NFA. In *Proceedings of FSTTCS*, pages 327–338, 2013.

20   Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

21   Domenic Quirl. Bachelor Thesis, supervised by Christof Löding, RWTH Aachen, 2018.