Nondeterminism in the Presence of a Diverse or Unknown Future [⋆]

Udi Boker[1], Denis Kuperberg[2], Orna Kupferman[2], and Michał Skrzypczak[3]

[1] IST Austria, Klosterneuburg, Austria.
[2] The Hebrew University, Jerusalem, Israel.
[3] University of Warsaw, Poland

Abstract. Choices made by nondeterministic word automata depend on both the past (the prefix of the word read so far) and the future (the suffix yet to be read). In several applications, most notably synthesis, the future is diverse or unknown, leading to algorithms that are based on deterministic automata. Hoping to retain some of the advantages of nondeterministic automata, researchers have studied restricted classes of nondeterministic automata. Three such classes are nondeterministic automata that are *good for trees* (GFT; i.e., ones that can be expanded to tree automata accepting the derived tree languages, thus whose choices should satisfy diverse futures), *good for games* (GFG; i.e., ones whose choices depend only on the past), and *determinizable by pruning* (DBP; i.e., ones that embody equivalent deterministic automata). The theoretical properties and relative merits of the different classes are still open, having vagueness on whether they really differ from deterministic automata. In particular, while DBP $\subseteq$ GFG $\subseteq$ GFT, it is not known whether every GFT automaton is GFG and whether every GFG automaton is DBP. Also open is the possible succinctness of GFG and GFT automata compared to deterministic automata. We study these problems for ω-regular automata with all common acceptance conditions. We show that GFT$=$GFG$\supset$DBP, and describe a determinization construction for GFG automata.

1 Introduction

Nondeterminism is very significant in word automata: it allows for exponential succinctness [14] and in some cases, such as Büchi automata, it also increases the expressive power [9]. In the automata-theoretic approach to formal verification, temporal logic formulas are translated to nondeterministic word automata [16]. In some applications, such as model checking, algorithms can proceed on the nondeterministic automaton, whereas in other applications, such as synthesis and control, they cannot. There, the advantages of nondeterminism are lost, and the algorithms involve a complicated determinization construction [15] or acrobatics for circumventing determinization [8].

To see the inherent difficulty of using nondeterminism in synthesis, let us review the current approach for solving the synthesis problem, going through games [4]. Let

[⋆] This work was supported in part by the Polish Ministry of Science grant no. N206 567840., Austrian Science Fund NFN RiSE (Rigorous Systems Engineering), ERC Advanced Grant QUAREM (Quantitative Reactive Modeling), and ERC Grant QUALITY.
The full version is available at the authors' URLs.

L be a language of infinite words over an alphabet $2^{I \cup O}$, where I and O are sets of input and output signals, respectively. The synthesis problem for L is to build a reactive system that outputs signals from 2^O upon receiving input signals from 2^I, such that the generated sequence (an infinite word over the alphabet $2^{I \cup O}$) is in L [12]. The problem is solved by taking a deterministic automaton $\mathcal{D}$ for L and conducting a two-player game on top of it. The players, "system" and "environment", generate words over $2^{I \cup O}$, where in each turn the environment first chooses the 2^I component of the next letter, the system responds with the 2^O component, and $\mathcal{D}$ moves to the successor state. The goal of the system is to generate an accepting run of $\mathcal{D}$ no matter which sequence of input assignments is generated by the environment. The system has a winning strategy iff the language L can be synthesized.

Now, if one tries to replace $\mathcal{D}$ with a nondeterministic automaton $\mathcal{A}$ for L, the system should also choose a transition to proceed with. Then, it might be that L is synthesizable and still the system has no winning strategy, as each choice of Σ may "cover" a strict subset of the possible futures.

Some nondeterministic automata are, however, good for games: in these automata it is possible to resolve the nondeterminism in a way that only depends on the past and still accepts all the words in the language. This notion, of *good for games* (GFG) automata was first introduced in [5].[1] Formally, a nondeterministic automaton over the alphabet Σ is GFG if there is a strategy that maps each word $x \in \Sigma^*$ to the transition to be taken after x is read. Note that a state q of the automaton may be reachable via different words, and the strategy may suggest different transitions from q after different words are read. Still, the strategy depends only on the past, meaning on the word read so far. Obviously, there exist GFG automata: deterministic ones, or nondeterministic ones that are *determinizable by pruning* (DBP); that is, ones that just add transitions on top of a deterministic automaton. In fact, these are the only examples known so far of GFG automata. [2] A natural question is whether all GFG automata are DBP.

More generally, a central question is what role nondeterminism can play in automata used for games, or abstractly put, in cases that the future is unknown. Specifically, can such nondeterminism add expressive power? Can it contribute to succinctness? Is it "real" or must it embody a deterministic choice?

Before addressing these questions, one should consider their tight connection to nondeterminism in tree automata for derived languages [7]: A nondeterministic word automaton $\mathcal{A}$ with language L is *good for trees* (GFT) if, when expanding its transition function to get a symmetric tree automaton, it recognizes the *derived language*, denoted $\mathrm{der}(L)$, of L; that is, all trees all of whose branches are in L [7]. Tree automata for derived languages were used for solving the synthesis problem [12] and are used when translating branching temporal logics such as CTL* to tree automata [3]. Analogously to GFG automata, the problem in using nondeterminism in GFT automata stems from the need to satisfy different futures (the different branches in the tree). For example,

[1] GFGness is also used in [2] in the framework of cost functions under the name "history-determinism".

[2] As explained in [5], the fact the GFG automata constructed there are DBP does not contradict their usefulness in practice, as their transition relation is simpler than the one of the embodied deterministic automaton and it can be defined symbolically.

solving the synthesis problem, the branches of the tree correspond to the possible input sequences, and when the automaton makes a guess, the guess has to be successful for all input sequences. The main difference between GFG and GFT is that the former can only use the past, whereas the latter can possibly take advantage of the future, except that the future is diverse.

A principal question is whether GFG and GFT automata are the same, meaning whether nondeterminism can take some advantage of a diverse future, or is it the same as only considering the past.

It is not difficult to answer all the above questions for safety languages; that is, when the language $L = L(\mathcal{A}) \subseteq \Sigma^\omega$ is such that all the words in L can be arranged in one tree. Then, a memoryless accepting run of $\mathcal{A}$ (that is, its expansion to a symmetric tree automaton for $\mathrm{der}(L)$) on this tree induces a deterministic automaton embodied in $\mathcal{A}$, meaning that $\mathcal{A}$ is DBP. Moving to general ω-regular languages, the first question, concerning expressiveness of deterministic versus GFT automata, was answered in [7] with respect to Büchi automata, and in [11] with respect to all levels of the Mostowski hierarchy. It is shown in these works that if $\mathrm{der}(L)$ can be recognized by a nondeterministic Büchi tree automaton, then L can be recognized by a deterministic Büchi word automaton, and similarly for parity conditions of a particular index. Thus, nondeterminism in the presence of unknown or diverse future does not add expressive power. The other questions, however, are open since the 90s.

In this paper we examine these questions further for automata with all common acceptance conditions. We first show that a Muller automaton is GFG iff it is GFT. As the Muller condition can describe all the common acceptance conditions (Büchi, co-Büchi, parity, Streett, and Rabin), the result follows to all of them. Intuitively, a GFT automaton $\mathcal{A}$ (or, equivalently, a nondeterministic tree automaton for a derived language) is limited in using information about the future, as different branches of the tree challenge it with different futures. Formally, we prove that $\mathcal{A}$ is GFG by using determinacy of a well-chosen game. The same game allows us to show that there is a deterministic automaton for $L(\mathcal{A})$ with the same acceptance condition as $\mathcal{A}$. This also simplifies the result of [11] and generalizes it to Muller conditions. Indeed, the proof in [11] is based on intricate arguments that heavily rely on the structure of parity condition.

Can GFG automata take some advantage of nondeterminism or do they simply hide determinism? We show the existence of GFG Büchi and co-Büchi automata that use the past in order to make decisions, and thus cannot have a memoryless strategy. Note that we use the basic acceptance conditions for these counter examples, thus the result follows to all common acceptance conditions. This is different from known results on GFG automata over finite words or weak GFG automata, where GFG automata are DBP [7, 10]. This result is quite surprising, as strategies in parity games are memoryless. We further build a GFG automaton that cannot be pruned into a deterministic automaton even with a finite unbounded look-ahead, meaning that even an unbounded yet finite view of the future cannot compensate on memorylessness.

Regarding succinctness, the currently known upper bound for the state blowup involved in determinizing a GFG parity automaton is exponential [7], with no nontrivial lower bound. We provide some insights on GFG automata, showing that in some cases its determinization is efficient. We show that if $\mathcal{A}$ and $\mathcal{B}$ are GFG Rabin automata that recognize a language L and its complement, then there is a deterministic Rabin automaton for L of size $|\mathcal{A} \times \mathcal{B}|$. Thus, in the context of GFG automata, determinization is essentially the same problem as complementation. Moreover, our construction shows that determinization cannot induce an exponential blowup both for an automaton and its complement. This is in contrast with standard nondeterminism, even over finite words. For example, both the language $L_k = (a+b)^* a(a+b)^k$ and its complement admit nondeterministic automata that are linear in k, while the deterministic ones are exponential in k.

Due to lack of space, some proofs are omitted, or shortened, and can be found in the full version.

2 Preliminaries

2.1 Trees and Labeled Trees

We consider trees over a set $\mathbb{D}$ of directions. A tree T is a prefix-closed subset of $\mathcal{T} = \mathbb{D}^*$. We refer to $\mathcal{T}$ as the *complete* $\mathbb{D}$*-tree*. The elements in T are called *nodes*, and ε is the *root* of T. For a node $u \in \mathbb{D}^*$ and $d \in \mathbb{D}$, the node ud is the *child* of u with *direction* d. A *path* of T is a set $\pi \subseteq T$, such that $\varepsilon \in \pi$ and for all $u \in \pi$, there is a unique $d \in \mathbb{D}$ with $ud \in \pi$. Note that each path π corresponds to an infinite word in $\mathbb{D}^\omega$.

For an alphabet Σ, a Σ*-labeled* $\mathbb{D}$*-tree* is a $\mathbb{D}$-tree in which each edge is labeled by a letter from Σ. We choose to label edges instead of nodes in order to be able to compose a set of words into a single tree, even when the set contains words that do not agree on their first letter. Formally, a Σ-labeled $\mathbb{D}$-tree is a pair $\langle T, t\rangle$ where $T \subseteq \mathcal{T}$ is a $\mathbb{D}$-tree and $t : T \setminus \{\varepsilon\} \to \Sigma$ labels each edge (or equivalently its target node) by a letter in Σ. Let $\mathcal{T}_{\mathbb{D},\Sigma}$ be the set of Σ-labeled $\mathbb{D}$-trees (not necessarily complete). We say that a word $w \in \Sigma^\omega$ is a *branch* of a tree $\langle T, t\rangle \in \mathcal{T}_{\mathbb{D},\Sigma}$ if there is a path $\pi = \{\varepsilon, u_1, u_2, \ldots\} \subseteq T$ such that $w = t(\pi) = t(u_1)t(u_2)\ldots$ We use $branches(\langle T, t\rangle)$ to denote the set of branches of $\langle T, t\rangle$. Note that $branches(\langle T, t\rangle)$ is a subset of Σ^ω.

2.2 Automata

Automata on words An automaton on infinite words is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha\rangle$, where Σ is the input alphabet, Q is a finite set of states, $q_0 \in Q$ is an (for simplicity, single) initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation such that $\langle q, a, q'\rangle \in \Delta$ if the automaton in state q, reading a, can move to state q'. The state $q_0 \in Q$ is the initial state, and α is an acceptance condition. Here we will use *Büchi*, *co-Büchi*, *parity*, *Rabin*, *Streett* and *Muller* automata. In a Büchi (resp. co-Büchi) conditions, $\alpha \subseteq Q$ is a set of accepting (resp. rejecting) states. In a parity condition of index $[i, j]$, the acceptance condition $\alpha : Q \to [i, j]$ is a function mapping each state to its priority (we use $[i, j]$ to denote the set $\{i, i+1, \ldots, j\}$). In a Rabin (resp. Streett) condition, $\alpha \subseteq 2^{2^Q \times 2^Q}$ is a set of pairs of sets of states, and in a Muller condition, $\alpha \subseteq 2^{2^Q}$ is a set of sets of states.

Since the transition relation may specify many possible transitions for each state and letter, the automaton $\mathcal{A}$ may be *nondeterministic*. If Δ is such that for every $q \in Q$ and $a \in \Sigma$, there is a single state $q' \in Q$ such that $\langle q, a, q'\rangle \in \Delta$, then $\mathcal{A}$ is a *deterministic* automaton.

Given an input word $w = a_0 \cdot a_1 \cdots$ in Σ^ω, a *run* of $\mathcal{A}$ on w is a function $r : \mathbb{N} \to Q$ where $r(0) = q_0$ and for every $i \geq 0$, we have $\langle r(i), a_i, r(i+1)\rangle \in \Delta$; i.e., the run starts in the initial state and obeys the transition function. For a run r, let $\inf(r)$ denote the set of states that r visits infinitely often. That is, $\inf(r) = \{q \in Q :$ for infinitely many $i \geq 0$, we have $r(i) = q\}$. The run r is *accepting* iff

- $\inf(r) \cap \alpha \neq \emptyset$, for a Büchi condition.
- $\inf(r) \cap \alpha = \emptyset$, for a co-Büchi condition.
- $\max\{\alpha(q) : q \in \inf(r)\}$ is even, for a parity condition.
- there exists $\langle E, F\rangle \in \alpha$, such that $\inf(r) \cap E = \emptyset$ and $\inf(r) \cap F \neq \emptyset$ for a Rabin condition.
- for all $\langle E, F\rangle \in \alpha$, we have $\inf(r) \cap E \neq \emptyset$ or $\inf(r) \cap F \neq \emptyset$ for a Streett condition.
- $\inf(r) \in \alpha$ for a Muller condition.

Note that Büchi and co-Büchi are dual, as well as Rabin and Streett. Parity and Muller are self-dual. Also note that Büchi and co-Büchi are a special case of parity, which is a special case of Rabin and Streett, which in turn are special cases of the Muller condition. An automaton $\mathcal{A}$ accepts an input word w iff there exists an accepting run of $\mathcal{A}$ on w. The *language* of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all words in Σ^ω that $\mathcal{A}$ accepts.

Automata on trees An automaton on Σ-labeled $\mathbb{D}$-trees is a tuple $\mathcal{A} = \langle \Sigma, \mathbb{D}, Q, q_0, \Delta, \alpha\rangle$, where Σ, Q, q_0, and α are as in automata on words, and $\Delta \subseteq Q \times (\Sigma \times Q)^{\mathbb{D}}$. Recall that we label the edges of the input trees. Accordingly, $\langle q, (a_d, q_d)_{d\in\mathbb{D}}\rangle \in \Delta$ if the automaton in state q, reading for each $d \in \mathbb{D}$ the letter a_d in direction d, can send a copy in q_d to the child in direction d. If for all $q \in Q$ and $(a_d)_{d\in\mathbb{D}} \in \Sigma^{\mathbb{D}}$, there is a single tuple $(q_d)_{d\in\mathbb{D}}$ such that $\langle q, (a_d, q_d)_{d\in\mathbb{D}}\rangle \in \Delta$, then $\mathcal{A}$ is deterministic.

A *run* of $\mathcal{A}$ on a Σ-labeled tree $\langle T, t\rangle$ is a function $r : T \to Q$ such that $r(\varepsilon) = q_0$ and for all $u \in T$, we have that $\langle r(u), (t(ud), r(ud))_{d\in\mathbb{D}}\rangle \in \Delta$. If for some directions d, the nodes ud are not in T, we assume that the requirement on them is satisfied. A run r on a tree T is accepting if the acceptance condition of the automaton is satisfied on all infinite paths of $\langle T, r\rangle$. For instance when $\mathcal{A}$ is a Büchi automaton, the run r is accepting if on all infinite paths in T it visits α infinitely often. As in automata on words, a tree $\langle T, t\rangle$ is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on $\langle T, t\rangle$, and the language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all trees in $\mathcal{T}_{\mathbb{D},\Sigma}$ that $\mathcal{A}$ accepts.

We use three letter acronyms in $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{F}, \mathrm{B}, \mathrm{C}, \mathrm{P}, \mathrm{R}, \mathrm{S}, \mathrm{M}\} \times \{\mathrm{W}, \mathrm{T}\}$ to denote classes of automata, with the first letter indicating whether this is a deterministic or nondeterministic automaton, the second whether it is an automaton on finite words or a Büchi / co-Büchi / parity / Rabin / Streett / Muller automaton, and the third whether it runs on words or trees. For example, a DBW is a deterministic Büchi automaton on infinite words.

2.3 Between Deterministic and Nondeterministic Automata

Let $L \subseteq \Sigma^\omega$ be a language of infinite words. We define the *derived language* of L, denoted $\mathrm{der}(L)$, as the set of Σ-labeled $\mathbb{D}$-trees all of whose branches are in L. Note that the definition has $\mathbb{D}$ as a parameter.

Since membership of a tree $\langle T, t\rangle$ in $\mathrm{der}(L)$ only depends on $branches(\langle T, t\rangle)$, we do not lose generality if we consider, in the context of derivable languages, trees in a *normal form* in which $\mathbb{D} = \Sigma$ and labels agree with directions. We note that examining trees for which $|\mathbb{D}| < |\Sigma|$ introduces an extra assumption on the set of possible futures, of which a nondeterministic automaton may take advantage.

Formally, we say that a Σ-labeled $\mathbb{D}$-tree $\langle T, t\rangle$ is in a normal form if $\Sigma = \mathbb{D}$, and for all $ua \in \Sigma^+$, we have $t(ua) = a$. Clearly, each Σ-labeled $\mathbb{D}$-tree $\langle T, t\rangle$ has a unique Σ-labeled Σ-tree $\langle T', t'\rangle$ in a normal form such that $branches(\langle T, t\rangle) = branches(\langle T', t'\rangle)$. Working with trees in a normal form enables us to identify the domain T with its labeling t. Thus, from now on we refer to a Σ-tree T, with the understanding that we talk about the unique Σ-labeled Σ-tree in normal form that has T as its underlying Σ-tree. For a Σ-tree T, the branch associated with a path $\{\epsilon, d_1, d_1d_2, d_1d_2d_3, \ldots\}$ is the infinite word $d_1d_2d_3\cdots$.The tree automata we consider also have $\mathbb{D} = \Sigma$ (and we omit $\mathbb{D}$ from the specification of the automaton).

Consider a nondeterministic word automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha\rangle$. Let $\mathcal{A}_t$ be the *expansion* of $\mathcal{A}$ to a tree automaton. Recall that we restrict attention to automata with $\mathbb{D} = \Sigma$. That is, $\mathcal{A}_t = \langle \Sigma, Q, q_0, \Delta_t, \alpha\rangle$ is such that for every $\langle q, (a_d, q_d)_{d\in\Sigma}\rangle \in Q \times (\Sigma \times Q)^{\Sigma}$, we have that $\langle q, (a_d, q_d)_{d\in\Sigma}\rangle \in \Delta_t$ iff for all $d \in \Sigma$, there is a transition $\langle q, a_d, q_d\rangle$ is in Δ. We say that $\mathcal{A}$ is *good for trees* (GFT, for short), if $L(\mathcal{A}_t) = \mathrm{der}(L(\mathcal{A}))$.

It is easy to see that when $\mathcal{A}$ is deterministic, then $\mathcal{A}$ is GFT. Indeed, $\mathcal{A}_t$ only accepts trees in $\mathrm{der}(L(\mathcal{A}))$, so $L(\mathcal{A}_t) \subseteq \mathrm{der}(L(\mathcal{A}))$. Conversely, since each prefix of a word in Σ^ω corresponds to a single prefix of a run of $\mathcal{A}$, we can compose the accepting runs of $\mathcal{A}$ of the words in $L(\mathcal{A})$ to an accepting run on $\mathcal{A}_t$ on every tree in $\mathrm{der}(L(\mathcal{A}))$.

General nondeterministic automata are not GFT. For example, let $\mathcal{A} = \langle \{a, b\}, \{q_0, q_1\}, q_0, \{\langle q_0, a, q_0\rangle, \langle q_0, b, q_0\rangle, \langle q_0, a, q_1\rangle, \langle q_1, a, q_1\rangle\}, \{q_1\}\rangle$ be the canonical NBW recognizing $L = (a+b)^*a^\omega$. Then, $\mathcal{A}_t$ cannot accept the tree $T = a^* \cup a^*ba^*$. Indeed, $\mathcal{A}_t$ has to move to q_1 at some point on the a^ω branch, but it then fails to accept other branches from that point, as there is no transition leaving q_1 labeled with b. In fact no NBT can recognize $\mathrm{der}(L(\mathcal{A}))$ [13].

A nondeterministic word automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha\rangle$ is *good for games* (GFG, for short) if there is a strategy $\sigma : \Sigma^* \to Q$ such that the following hold: (1) The strategy σ is compatible with Δ. That is, for all $\langle u, a\rangle \in \Sigma^* \times \Sigma$, we have $\langle \sigma(u), a, \sigma(ua)\rangle \in \Delta$. (2) The restriction imposed by σ does not exclude words from $L(\mathcal{A})$; that is, for all $u = u_0 \cdot u_1 \cdot u_2, \cdots \in L(\mathcal{A})$, the sequence $\sigma(\varepsilon)$, $\sigma(u_0)$, $\sigma(u_0u_1)$, $\sigma(u_0u_1u_2), \ldots$ satisfies the acceptance condition α.

Finally, Σ is *determinizable by pruning* (DBP, for short) if it can be determinized to an equivalent automaton by removing some of its transitions.

A DBP automaton is obviously also GFG, using the strategy that follows the unpruned transitions. A GFG automaton is also GFT, as the latter can resolve its nondeterminism using the strategy that witnesses the GFGness.

Proposition 1. *If an automaton $\mathcal{A}$ is DBP, then it is GFG. If $\mathcal{A}$ is GFG then $\mathcal{A}$ is GFT.*

Let $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha\rangle$ be a tree automaton. The word automaton associated with $\mathcal{A}$ is $\mathcal{A}_w = \langle \Sigma, Q, q_0, \Delta_w, \alpha\rangle$, where Δ_w is such that $\langle q, a, q'\rangle \in \Delta_w$ iff Δ

has a transition from q in which q' is sent to some direction along an edge labeled a. Formally, there is a transition $\langle q, (a_d, q_d)_{d \in \Sigma} \rangle \in \Delta$ with $(a_d, q_q) = (a, q')$ for some $d \in \Sigma$. It is easy to see that $\mathcal{A}_w$ accepts exactly all infinite words that appear as a branch of some tree accepted by $\mathcal{A}$. Note that if $L(\mathcal{A}) = \mathrm{der}(L)$, then $L(\mathcal{A}_w) = L$, and $L((\mathcal{A}_w)_t) = \mathrm{der}(L)$, so $\mathcal{A}_w$ is GFT.

3 From GFT to GFG

In this section we prove that if an NMW is GFT then it is also GFG. In addition, we show that GFG automata admit finite memory strategies and we study connections with [11].

The crucial tool in the proof is the following infinite-duration perfect-information game between two players $\exists$ and $\forall$. Let $\mathcal{A} = \langle \mathbb{D}, Q^{\mathcal{A}}, q_I^{\mathcal{A}}, \Delta^{\mathcal{A}}, \alpha^{\mathcal{A}} \rangle$ be an arbitrary NMW. Let $\mathcal{D} = \langle \mathbb{D}, Q^{\mathcal{D}}, q_I^{\mathcal{D}}, \Delta^{\mathcal{D}}, \alpha^{\mathcal{D}} \rangle$ be a DSW recognizing $L(\mathcal{A})$. The arena of the game $\mathcal{G}(\mathcal{A})$ is $Q^{\mathcal{A}} \times Q^{\mathcal{D}}$ and its initial position $\langle q_0, p_0 \rangle$ is the pair of initial states $(q_I^{\mathcal{A}}, q_I^{\mathcal{D}})$. In the i-th round of a play, $\forall$ chooses a letter $d_i \in \mathbb{D}$ and $\exists$ chooses a state q_{i+1} such that $\langle q_i, d_i, q_{i+1} \rangle \in \Delta^{\mathcal{A}}$. The successive position is (q_{i+1}, p_{i+1}), where p_{i+1} is the unique state of $\mathcal{D}$ such that $\langle p_i, d_i, p_{i+1} \rangle \in \Delta^{\mathcal{D}}$.

An infinite play $\Pi = (\langle q_0, p_0 \rangle, d_0), (\langle q_1, p_1 \rangle, d_1), \ldots$ is won by $\exists$ if either the run $\Pi_{\mathcal{A}} := (q_i)_{i \in \mathbb{N}}$ is accepting or the run $\Pi_{\mathcal{D}} := (p_i)_{i \in \mathbb{N}}$ is rejecting. Note that since $\mathcal{D}$ recognizes $L(\mathcal{A})$, it follows that $\Pi_{\mathcal{D}}$ is rejecting iff $\Pi_{\mathbb{D}} := (d_i)_{i \in \mathbb{N}}$ does not belong to $L(\mathcal{A})$.

Since the game is ω-regular, it admits finite-memory winning strategies. The winning condition for $\exists$ in $\mathcal{G}(\mathcal{A})$ is the disjunction of $\alpha^{\mathcal{A}}$ with the Rabin condition that is dual to $\alpha^{\mathcal{D}}$. In particular, when $\mathcal{A}$ is an NRW, then the winning condition is a Rabin condition, thus if $\exists$ has a winning strategy in $\mathcal{G}(\mathcal{A})$, she also has a memoryless one.

Obviously, a strategy for $\exists$ is a strategy for resolving the nondeterminism in $\mathcal{A}$. Hence, we have the following.

Lemma 1. *If $\exists$ has a winning strategy in $\mathcal{G}(\mathcal{A})$ then $\mathcal{A}$ is GFG. Additionally, there exists a finite-memory strategy σ witnessing its GFGness. If $\mathcal{A}$ is an NRW (NMW), then σ is at most exponential (resp. doubly exponential) in the size of $\mathcal{A}$.*

Lemma 2. *If $\forall$ has a winning strategy in $\mathcal{G}(\mathcal{A})$, then $\mathcal{A}$ is not GFT.*

Proof. Let $\sigma_\forall \colon (Q^{\mathcal{A}})^+ \to \mathbb{D}$ be a winning strategy of $\forall$ in $\mathcal{G}(\mathcal{A})$. Thus, for a sequence $\boldsymbol{q}$ of states, namely the history of the game so far, the strategy $\sigma_\forall$ assigns the letter to be played by $\forall$. Note that for some sequences $\boldsymbol{q} \in \left(Q^{\mathcal{A}}\right)^+$, the value $\sigma_\forall(\boldsymbol{q})$ is set arbitrarily, as there is no play corresponding to such a sequence (e.g., if $q_0 \neq q_I^{\mathcal{A}}$).

Let $u \in \mathbb{D}^*$ be a word and $\boldsymbol{q} = (q_0, \ldots, q_{j-1}) \in (Q^{\mathcal{A}})^+$ be a sequence of states. We say that $\boldsymbol{q}$ *forces* u if $(\boldsymbol{q}, u)$ is a prefix of a play in $\mathcal{G}(\mathcal{A})$ in which $\forall$ plays according to the strategy $\sigma_\forall$. Formally, $\boldsymbol{q}$ forces u if the following hold: (1) $|u| = |\boldsymbol{q}| = j > 0$, (2) $q_0 = q_I^{\mathcal{A}}$, (3) for every $i < j - 1$, the tuple $(q_i, u(i), q_{i+1})$ is a transition of $\mathcal{A}$, and (4) for every $i < j$, the letter $u(i)$ equals $\sigma_\forall(q_0 \ldots q_i)$.

Let $T \subseteq \mathbb{D}^*$ be the set of words $u \in \mathbb{D}^*$ such that there is a sequence $\boldsymbol{q} \in \left(Q^{\mathcal{A}}\right)^*$ that forces u. Note that T is prefix-closed, so it is a $\mathbb{D}$-branching tree. We first show that $T \in \mathrm{der}(L(\mathcal{A}))$. Consider an infinite path π of T. Let $\pi = \{\epsilon, u_1, u_2, \ldots\}$ and let $(\boldsymbol{q}_i)_{i>0}$ be sequences $\boldsymbol{q}_i \in \left(Q^{\mathcal{A}}\right)^*$ such that $\boldsymbol{q}_i$ forces u_i. Note that $|\boldsymbol{q}_i| = |u_i| = i$. Since there are finitely many states in $\mathcal{A}$, there exists a subsequence of $(\boldsymbol{q}_i)_{i>0}$ that is pointwise convergent to a limit $\rho \in (Q^{\mathcal{A}})^{\omega}$. For instance, this sequence can be built by iteratively choosing states that appear in infinitely many of the $\boldsymbol{q}_i$. For a finite or infinite sequence of states $\boldsymbol{q}$ and an index $j \in \mathbb{N}$, let $\boldsymbol{q}_{|j}$ be the prefix of $\boldsymbol{q}$ of length j. It follows that for every $j \in \mathbb{N}$ there exists $i > 0$ such that $\rho_{|j} = (\boldsymbol{q}_i)_{|j}$.

Let Π be the play that is the outcome of $\forall$ playing $\sigma_\forall$ and $\exists$ playing successive states of ρ. By the above, for every $j \in \mathbb{N}$, we have $\langle \rho(j), \pi(j), \rho(j+1) \rangle \in \Delta^{\mathcal{A}}$. Therefore, the play Π is well defined, $\Pi_{\mathbb{D}} = \pi$, and $\Pi_{\mathcal{A}} = \rho$. Since $\sigma_\forall$ is a winning strategy, $\Pi_{\mathcal{D}}$ is accepting, and therefore $\pi \in L(\mathcal{A})$. Since we showed the above for all paths π of T, we conclude that $T \in \mathrm{der}(L(\mathcal{A}))$.

Assume now, by way of contradiction, that $\mathcal{A}$ is GFT. Thus, $L(\mathcal{A}_t) = \mathrm{der}(L(\mathcal{A}))$. Since $T \in \mathrm{der}(L(\mathcal{A}))$, there is an accepting run ρ_t of $\mathcal{A}_t$ on T. Let Π be the infinite play of $\mathcal{G}(\mathcal{A})$ that is the outcome of $\forall$ playing $\sigma_\forall$ and $\exists$ playing transitions of ρ_t: if $\forall$ played $u \in \mathbb{D}^+$, then $\exists$ plays $\rho_t(u)$. Since ρ_t is accepting, $\Pi_{\mathcal{A}}$ is also accepting, and $\exists$ wins Π, contradicting the fact that $\forall$ plays his winning strategy. ☐

Observe that the arena of the game $\mathcal{G}(\mathcal{A})$ is finite and the winning condition for $\exists$ is ω-regular. Thus, the game is determined (see [1,4]) and one of the conditions in Lemma 1 or 2 holds. Hence $\mathcal{A}$ is either GFG or not GFT, and we can conclude with the following:

Theorem 1. *If an NMW is GFT then it is GFG. Moreover, there exists a finite-memory strategy σ witnessing its GFGness. If $\mathcal{A}$ is an NRW (NMW), then σ is at most exponential (resp. doubly exponential) in the size of $\mathcal{A}$.*

The following observation can be seen as an extension of [11] from parity condition to general Muller acceptance conditions. The only difference here is that we work with Σ-labelled $\mathbb{D}$-trees with $|\mathbb{D}| \geq |\Sigma|$, while [11] was working on binary trees with arbitrary alphabets. Again, we believe that these differences in the formalisms do not reflect essential behaviors of automata on infinite trees, since a simple encoding always allows to go from one formalism to another. Notice that the proof in [11] relies crucially on the structure of parity conditions and does not seem to generalize to arbitrary Muller conditions. In the following statement we use γ to denote an acceptance condition, e.g. a parity $[i, j]$ condition, a Rabin condition with k pairs, a Muller condition with k sets, etc.

Corollary 1. *Consider an ω-regular word language L. If $\mathrm{der}(L)$ can be recognized by a nondeterministic γ tree automaton, then L can be recognized by a deterministic γ word automaton.*

Proof. The word automaton $\mathcal{A}_w = \langle \mathbb{D}, Q, q_I, \Delta, \alpha \rangle$ associated with $\mathcal{A}$ is GFT, so by Theorem 1 it is GFG. By Theorem 1, the fact that $\mathcal{A}_w$ is GFG is witnessed by a strategy σ, using a finite memory structure M with an initial state $m_0 \in M$. That is to say, $\sigma \colon Q \times M \times \mathbb{D} \to Q \times M$ can be used to guide choices in $\mathcal{A}_w$, ensuring that all words in L are accepted. Therefore, we can build the required deterministic automaton $\mathcal{D}$ with states $Q \times M$, where the transition function maps a state $\langle q, m \rangle$ and letter $d \in \mathbb{D}$ to the state $\sigma(q, m, d)$. The acceptance condition of $\mathcal{D}$ is identical to α, and needs only to consider the Q-components of the states (the M component does not play a role for acceptance), and is thus of type γ. Since an accepting run of $\mathcal{D}$ induces an accepting run of $\mathcal{A}_w$, we have $L(\mathcal{D}) \subseteq L$. Conversely, if π is a word in L, the unique run of $\mathcal{D}$ on π corresponds to the execution of the GFG strategy σ in $\mathcal{A}_w$, and it thus accepting. Hence, $L(\mathcal{D}) = L$, for the deterministic γ automaton $\mathcal{D}$.

$\square$

Observe that since deterministic automata are clearly GFT, the other direction of Corollary 1 is trivial.

4 From GFG to Deterministic Automata

In this section we study determinization of GFG automata. As discussed in Section 2, every DBP automaton (that is, a nondeterministic automaton that is determinizable by pruning) is GFG. The first question we consider is whether the converse is also valid, thus whether every GFG is DBP. We show that, surprisingly, not all GFG Büchi and co-Büchi automata are DBP. Note that since these counter examples are with basic acceptance conditions, the result follows for all common acceptance conditions. This gives rise to a second question, of the blowup involved in determinizing GFG NRWs. We describe a determinization construction that generates a DRW whose size is bounded by the product of the input GFG NRW and a GFG NRW for its complement.

4.1 GFG Büchi and co-Büchi Automata are not DBP

There is a strong intuition that a GFG NBW can be determinized by pruning: By definition, the choices of a GFG NBW are independent on the future. Accordingly, the question about GFG NBWs being DBP amounts to asking whether the choices are independent of the past. Since Büchi games are memoryless, it is tempting to believe that the answer is positive. A positive answer is also supported by the fact that GFG NBWs that recognize safety languages are DBP. In addition, all GFG NBWs studied so far, and in particular these constructed in [5] are DBP. Yet, as we show in this section, GFG NBWs are not DBP.

We start with a simple meta-NBW that operates over infinite words composed of two finite words (tokens), x and y. Afterwards, we formalize it to a GFG NBW. Moreover, we show that even more flexible versions of DBP, such as ones that allow the "deterministic" automata to have finite look-ahead are not sufficient.

A meta example. The meta-NBW $\mathcal{M}$, described in Figure 1, accepts exactly all words that contain infinitely many 'xx's or 'yy's. That is, $L(\mathcal{M}) = [(x+y)^*(xx+yy)]^\omega$. It is not hard to see that $\mathcal{M}$ is GFG by using the following strategy in its single nondeterministic state q_0: "if the last token was x then go to q_1 else go to q_2". On the other hand, determinizing q_0 to always choose q_1 loses y^ω and always choosing q_2 loses x^ω. Hence, $\mathcal{M}$ is not DBP.

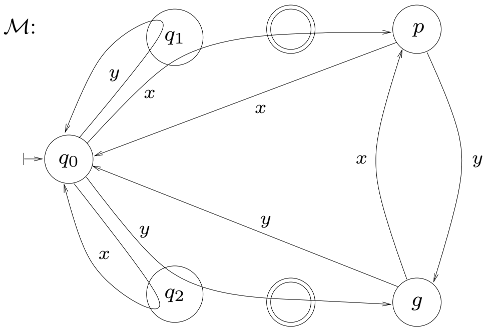

Fig. 1. A meta GFG NBW that is not DBP.

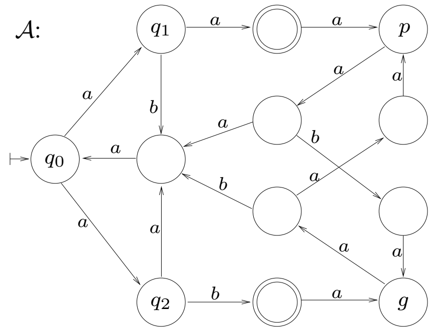

Fig. 2. A GFG NBW that is not DBP.

A concrete example. Using the above meta-NBW with $x = aaa$ and $y = aba$ provides the NBW $\mathcal{A}$, described in Figure 2, whose language is $L = [(aaa + aba)^*(aaa\,aaa + aba\,aba)]^\omega$. Essentially, it follows from the simple observations that $\mathcal{A}$ has an infinite run on a word w iff $w \in (aaa + aba)^\omega$. Also, after a prefix whose length divides by 3, a run of $\mathcal{A}$ can only be in either q_0, p or g.

A co-Büchi example. In order to show that these counter examples are not specific to the Büchi condition, we give another example of GFG which is not DBP, using the co-Büchi condition. For simplicity, the acceptance is now specified via the transitions instead of the states. Dashed transitions are co-Büchi, i.e. accepting runs must take them only finitely often. (It is not hard to build a counter-example with co-Büchi condition on states from this automaton.)

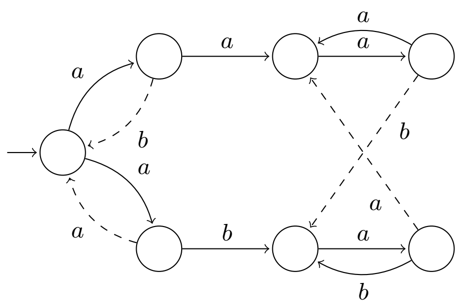

Fig. 3. A co-Büchi automaton that recognizes the language $(aa + ab)^*[a^\omega + (ab)^\omega]$. It is GFG but not DBP. Note that unlike the Büchi counter-example, one good choice is enough for getting an accepting run.

Theorem 2. *GFG NPWs are not DBP, even for Büchi and co-Büchi conditions.*

Proof. We prove that the NBW $\mathcal{A}$ from Figure 2 is GFG and is not DBP. First, the only nondeterminism of $\mathcal{A}$ is in q_0. The following strategy, applied in q_0, witnesses that $\mathcal{A}$ is

GFG: “if the last three letters were ‘aaa’ then go to q_1 else go to q_2”. Now, to see that $\mathcal{A}$ is not DBP, recall that the only nondeterminism of $\mathcal{A}$ is in q_0. Therefore, there are two possible prunings to consider: the DBW $\mathcal{A}'$ in which $\delta(q_0, a) = q_1$ and the DBW $\mathcal{A}''$ in which $\delta(q_0, a) = q_2$. With the former, $(aba)^\omega \in L(\mathcal{A}) \setminus L(\mathcal{A}')$ and with the latter $(aaa)^\omega \in L(\mathcal{A}) \setminus L(\mathcal{A}'')$. □

While the GFG NBW $\mathcal{A}$ used in the proof of Theorem 2 is not DBP, it can be determinized by merging the states q_1 and q_2, to which q_0 goes nondeterministically, and then pruning. Furthermore, $\mathcal{A}$ is “almost deterministic”, in the sense that a look-ahead of one letter into the future is sufficient for resolving its nondeterminism. One may wonder whether GFG NBWs are determinizable with more flexible definitions of pruning. We answer this to the negative, describing (in the full version) a GFG NBW in which merging the target states of the nondeterminism cannot help, and no finite look-ahead suffices for resolving the nondeterminism.

Theorem 3. *There are GFG NBWs that cannot be pruned into deterministic automata with unbounded yet finite look-ahead, or by merging concurrent target states.*

4.2 A Determinization Construction

In this section, we show that determinization in the context of GFG automata cannot induce an exponential blowup for both a language and its complement. This gives a serious hint towards the fact that determinization is simpler in the case of GFG. Indeed, for general nondeterministic automata, the blowup can occur on both sides. For example, consider the family of languages of finite words $L_k = (a + b)^* a (a + b)^k$. While for all $k \geq 1$, both L_k and its complement have nondeterministic automata with $O(k)$ states, a deterministic automaton for L_k must have at least 2^k states.

We now assume that we have a Rabin GFG (NRW-GFG) automaton for L, and an NRW-GFG for the complement $\mathrm{comp}(L)$ of L. We show the following.

Theorem 4. *If $\mathcal{A}$ is an NRW-GFG for L with n states, and $\mathcal{B}$ is an NRW-GFG for $\mathrm{comp}(L)$ with m states, then we can build a DRW for L with nm states.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta_\mathcal{A}, \alpha \rangle$ be an NRW-GFG for L, and $\mathcal{B} = \langle \Sigma, P, p_0, \Delta_\mathcal{B}, \beta \rangle$ be an NRW-GFG for $\mathrm{comp}(L)$. We construct a Rabin game $\mathcal{G}$ between two players, $\exists$ and $\forall$, as follows. The arena of $\mathcal{G}$ is the product $\mathcal{A} \times \mathcal{B}$. Formally, the positions of the game are pairs of states $(q, p) \in Q \times P$, and there is an edge $(q, p) \xrightarrow{a} (q', p')$ in $\mathcal{G}$ if $(q, a, q') \in \Delta_\mathcal{A}$ and $(p, a, p') \in \Delta_\mathcal{B}$. The initial position of the game is (q_0, p_0).

A turn from position (p, q) is played as follows: First, $\forall$ chooses a letter a in Σ. Then, $\exists$ chooses an edge $(q, p) \xrightarrow{a} (q', p')$. The game then continues from (q', p'). Thus, the outcome of a play is an infinite sequence $\pi = (q_0, p_0), (q_1, p_1), (q_2, p_2), \ldots$ of positions. Note that π combines the run $\pi_\mathcal{A} = q_0, q_1, q_2, \ldots$ of $\mathcal{A}$ and the run $\pi_\mathcal{B} = p_0, p_1, p_2, \ldots$ of $\mathcal{B}$.

The winning condition for $\exists$ is that either $\pi_\mathcal{A}$ satisfies α or $\pi_\mathcal{B}$ satisfies β. These objectives can be easily specified by a Rabin winning condition. It is easy to see that $\exists$ has a winning strategy: it suffices to play in both automata according to their respecting GFG strategies. By definition of GFG automata, if $\forall$ generates a word u in L, the run

$\pi_{\mathcal{A}}$ is accepting in $\mathcal{A}$ and thus satisfies α. Likewise, if $u \in \mathrm{comp}(L)$, then the run $\pi_{\mathcal{B}}$ is accepting in $\mathcal{B}$ and thus satisfies β. Since every word is either in L or in $\mathrm{comp}(L)$, the winning condition for $\exists$ is always satisfied.

It is known that Rabin games admit memoryless strategies [6]. Hence, $\exists$ actually has a memoryless winning strategy in $\mathcal{G}$. Such a strategy maps each position $(q, p) \in Q \times P$ and letter $a \in \Sigma$ to a destination (q', p'). Hence, by keeping only edges used by the memoryless strategy, we can prune the nondeterministic product automaton $\mathcal{A} \times \mathcal{B}$ into a deterministic automaton that accepts all words in Σ^ω. Moreover, by simply forgetting the acceptance condition of $\mathcal{B}$, and keeping only the one from $\mathcal{A}$, we get a DRW $\mathcal{D}$ recognizing L. Notice that if $\mathcal{A}$ was for instance Büchi, or parity with index $[i, j]$, the automaton $\mathcal{D}$ has the same acceptance condition. The number of states of $\mathcal{D}$ is $|P \times Q|$. $\square$

References

1. J.R. Büchi and L.H. Landweber. *Solving Sequential Conditions by Finite State Strategies*. CSD TR. 1967.
2. T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.
3. E.A. Emerson and A. P. Sistla. Deciding branching time logic. In *Proc. 16th ACM Symp. on Theory of Computing*, pages 14–24, 1984.
4. E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
5. T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.
6. N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure Appl. Logic*, 69(2-3):243–268, 1994.
7. O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.
8. O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.
9. L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.
10. G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.
11. D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th Symp. on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*. Springer, 1998.
12. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.
13. M.O. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.
14. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.
15. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.
16. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.