How Deterministic are Good-For-Games Automata?

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

[1]Interdisciplinary Center, Herzliya, Israel*
[2]The Hebrew University, Israel†
[3]University of Warsaw, Poland‡

Abstract

In *good for games* (GFG) automata, it is possible to resolve nondeterminism in a way that only depends on the past and still accepts all the words in the language. The motivation for GFG automata comes from their adequacy for games and synthesis, wherein general nondeterminism is inappropriate. We continue the ongoing effort of studying the power of nondeterminism in GFG automata. Initial indications have hinted that every GFG automaton embodies a deterministic one. Today we know that this is not the case, and in fact GFG automata may be exponentially more succinct than deterministic ones.

We focus on the *typeness* question, namely the question of whether a GFG automaton with a certain acceptance condition has an equivalent GFG automaton with a weaker acceptance condition on the same structure. Beyond the theoretical interest in studying typeness, its existence implies efficient translations among different acceptance conditions. This practical issue is of special interest in the context of games, where the Büchi and co-Büchi conditions admit memoryless strategies for both players. Typeness is known to hold for deterministic automata and not to hold for general nondeterministic automata.

We show that GFG automata enjoy the benefits of typeness, similarly to the case of deterministic automata. In particular, when Rabin or Streett GFG automata have equivalent Büchi or co-Büchi GFG automata, respectively, then such equivalent automata can be defined on a substructure of the original automata. Using our typeness results, we further study the place of GFG automata in between deterministic and nondeterministic ones. Specifically, considering automata complementation, we show that GFG automata lean toward nondeterministic ones, admitting an exponential state blow-up in the complementation of a Streett automaton into a Rabin automaton, as opposed to the constant blow-up in the deterministic case.

*This research was supported by the Israel Science Foundation, grant no. 1373/16.

†This research has received funding from the European Research Council under the EU's 7-th Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410.

‡Supported by the Polish National Science Centre (decision UMO-2016/21/D/ST6/00491).

1 Introduction

Nondeterminism is a prime notion in theoretical computer science. It allows a computing machine to examine, in a concurrent manner, all its possible runs on a certain input. For automata on finite words, nondeterminism does not increase the expressive power, yet it leads to an exponential succinctness [15]. For automata on infinite words, nondeterminism may increase the expressive power and also leads to an exponential succinctness. For example, nondeterministic Büchi automata are strictly more expressive than their deterministic counterpart [11]. In the automata-theoretic approach to formal verification, we use automata on infinite words in order to model systems and their specifications. In particular, temporal logic formulas are translated to nondeterministic word automata [19]. 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 [16] or acrobatics for circumventing determinization [10]. Essentially, the inherent difficulty of using nondeterminism in synthesis lies in the fact that each guess of the nondeterministic automaton should accommodate all 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 while still accepting all the words in the language. This notion, of *good for games* (GFG) automata was first introduced in [4].[1] Formally, a nondeterministic automaton $\mathcal{A}$ over an alphabet Σ is GFG if there is a strategy g that maps each finite word $u \in \Sigma^+$ to the transition to be taken after u is read; and following g results in accepting all the words in the language of $\mathcal{A}$. Note that a state q of $\mathcal{A}$ may be reachable via different words, and g may suggest different transitions from q after different words are read. Still, g depends only on the past, namely on the word read so far. Obviously, there exist GFG automata: deterministic ones, or nondeterministic ones that are *determinizable by pruning* (DetByP); that is, ones that just add transitions on top of a deterministic automaton. In fact, the GFG automata constructed in [4] are DetByP.[2]

Our work continues a series of works that have studied GFG automata: their expressive power, succinctness, and constructions for them, where the key challenge is to understand the power of nondeterminism in GFG automata. Let us first survey the results known so far. In terms of expressive power, it is shown in [8, 14] that GFG automata with an acceptance condition of type γ (e.g., Büchi) are as expressive as deterministic γ automata.[3] Thus, as far as expressiveness is

[1] GFGness is also used in [3] in the framework of cost functions under the name "history-determinism".

[2] As explained in [4], the fact that the GFG automata constructed there are DetByP 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.

[3] The results in [8, 14] are given by means of *tree automata for derived languages*, yet, by [2], the results hold also for GFG automata.

concerned, GFG automata behave like deterministic ones. The picture in terms of succinctness is diverse. For automata on finite words, GFG automata are always DetByP [8, 12]. For automata on infinite words, in particular Büchi and co-Büchi automata[4], GFG automata need not be DetByP [2]. Moreover, the best known determinization construction of GFG Büchi automata is quadratic, whereas determinization of GFG co-Büchi automata has an exponential blow-up lower bound [6]. Thus, in terms of succinctness, GFG automata on infinite words are more succinct (possibly even exponentially) than deterministic ones.

For deterministic automata, where Büchi and co-Büchi automata are less expressive than Rabin and Streett ones, researchers have come up with the notion of an automaton being *type* [5]. Consider a deterministic automaton $\mathcal{A}$ with acceptance condition of type γ and assume that $\mathcal{A}$ recognizes a language that can be recognized by some deterministic automaton with an acceptance condition of type β that is weaker than γ. When deterministic γ automata are β-type, it is guaranteed that a deterministic β-automaton for the language of $\mathcal{A}$ can be defined on top of the structure of $\mathcal{A}$. For example, deterministic Rabin automata being Büchi-type [5] means that if a deterministic Rabin automaton $\mathcal{A}$ recognizes a language that can be recognized by a deterministic Büchi automaton, then $\mathcal{A}$ has an equivalent deterministic Büchi automaton on the same structure. Thus, the basic motivation of typeness is to allow simplifications of the acceptance conditions of the considered automata without complicating their structure. Applications of this notion are much wider [5]. In particular, in the context of games, the Büchi and co-Büchi conditions admit memoryless strategies for both players, which is not the case for the Rabin and Streett conditions [18]. Thus, the study of typeness in the context of GFG automata addresses also the question of simplifying the memory requirements of the players. In addition, as we elaborate in Section 7, it leads to new and non-trivial bounds on the blow-up of transformations between GFG automata and their complementation.

Recall that deterministic Rabin automata are Büchi-type. Dually, deterministic Streett automata are co-Büchi-type. Typeness can be defined also with respect to nondeterministic automata, yet it crucially depends on the fact that the automaton is deterministic. Indeed, nondeterministic Rabin are not Büchi-type. Even with the co-Büchi acceptance condition, where nondeterministic co-Büchi automata recognize only a subset of the ω-regular languages, nondeterministic Streett automata are not co-Büchi-type [7].

We first show that typeness is strongly related with determinism even when slightly relaxing the typeness notion to require the existence of an equivalent automaton on a substructure of the original automaton, instead of on the exact original structure, and even when we restrict attention to an *unambiguous* automaton, namely one that has a single accepting run on each word in its language. We describe an unambiguous parity automaton $\mathcal{A}$, such that its language is recognized by a deterministic Büchi automaton, yet it is impossible to define a Büchi acceptance condition on top of a substructure of $\mathcal{A}$. We also point to

[4]See Section 2.1 for the full definition of the various acceptance conditions.

a dual result in [7], with respect to the co-Büchi condition, and observe that it applies also to the relaxed typeness notion.

We then show that for GFG automata, typeness, in its relaxed form, does hold. Notice that once considering GFG automata with no redundant transitions, which we call *tight*, the two typeness notions coincide. Obviously, all GFG automata can be tightened by removal of redundant transitions (Lemma 2.4). In particular, we show that the typeness picture in GFG automata coincides with the one in deterministic automata: Rabin GFG automata are Büchi type, Streett GFG automata are co-Büchi type, and all GFG automata are type with respect to the weak acceptance condition. Unlike the deterministic case, however, the Rabin case is not a simple dualization of the Streett case; it is much harder to prove and it requires a stronger notion of tightness.

We continue with using our typeness results for further studying the place of GFG automata in between deterministic and nondeterministic ones. We start with showing that all GFG automata that recognize languages that can be defined by deterministic weak automata are DetByP. This generalizes similar results about safe and co-safe languages [7]. We then show that all unambiguous GFG automata are also DetByP. Considering complementation, GFG automata lie in between the deterministic and nondeterministic settings—the complementation of a Büchi automaton into a co-Büchi automaton is polynomial, as is the case with deterministic automata, while the complementation of a co-Büchi automaton into a Büchi automaton as well as the complementation of a Streett automaton into a Rabin automaton is exponential, as opposed to the constant blow-up in the deterministic case. We conclude with proving a doubly-exponential lower bound for the translation of LTL into GFG automata, as is the case with deterministic automata.

The paper is structured as follows. In Section 2 we provide the relevant notions about languages and GFG automata. Section 3 contains examples showing that typeness does not hold for the case of unambiguous automata. The next three sections, Sections 4, 5, and 6, provide the main positive results of this work: co-Büchi typeness for GFG-Streett; Büchi typeness for GFG-Rabin; and weak typeness for GFG-Büchi and GFG-co-Büchi, respectively. Finally, in Section 7 we continue to study the power of nondeterminism in GFG automata, looking into automata complementation and translations of LTL formulas to GFG automata.

2 Preliminaries

2.1 Automata

An automaton on infinite words is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is an input alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta \colon Q \times \Sigma \to 2^Q$ is a transition function that maps a state and a letter to a set of possible successors, and α is an acceptance condition. The first four elements, namely $\langle \Sigma, Q, \delta, Q_0 \rangle$, are the automaton's *structure*. We consider here the *Büchi*,

co-Büchi, *parity*, *Rabin*, and *Streett* acceptance conditions. (The *weak* condition is defined in Section 6.) In Büchi, and co-Büchi conditions, $\alpha \subseteq Q$ is a set of states. In a parity condition, $\alpha \colon Q \to \{0, \ldots, k\}$ is a function mapping each state to its priority. In a Rabin and Streett conditions, $\alpha \subseteq 2^{2^Q \times 2^Q}$ is a set of pairs of sets of states. The *index* of a Rabin or Streett condition is the number of pairs in it. For a state q of $\mathcal{A}$, we denote by $\mathcal{A}^q$ the automaton that is derived from $\mathcal{A}$ by changing the set of initial states to $\{q\}$. A *transition* of $\mathcal{A}$ is a triple $\langle q, a, q' \rangle$ such that $q' \in \delta(q, a)$. We extend δ to sets of states and to finite words in the expected way. Thus, for a set $S \subseteq Q$, a letter $a \in \Sigma$, and a finite word $u \in \Sigma^*$, we have that $\delta(S, \epsilon) = S$, $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$, and $\delta(S, u \cdot a) = \delta(\delta(S, u), a)$. Then, we denote by $\mathcal{A}(u)$ the set of states that $\mathcal{A}$ may reach when reading u. Thus, $\mathcal{A}(u) = \delta(Q_0, u)$.

Since the set of initial states need not be a singleton and the transition function may specify several successors for each state and letter, the automaton $\mathcal{A}$ may be *nondeterministic*. If $|Q_0| = 1$ and $|\delta(q, a)| \leq 1$ for every $q \in Q$ and $a \in \Sigma$, then $\mathcal{A}$ is *deterministic*.

Given an input word $w = a_1 \cdot a_2 \cdots$ in Σ^ω, a *run* of $\mathcal{A}$ on w is an infinite sequence $r = r_0, r_1, r_2, \ldots \in Q^\omega$ such that $r_0 \in Q_0$ and for every $i \geq 0$, we have $r_{i+1} \in \delta(r_i, a_{i+1})$; 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, $\inf(r) = \{q \in Q \mid \text{for infinitely many } i \geq 0, \text{ we have } r_i = q\}$.

A set of states S satisfies an acceptance condition α (or *is accepting*) iff

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

Notice that Büchi and co-Büchi are dual, and so are Rabin and Streett. Also note that the Büchi and co-Büchi conditions are special cases of parity, which is a special case of Rabin and Streett. In the latter conditions, we refer to the sets E and F as the "bad" and "good" sets, respectively. Finally, note that a Rabin pair may have an empty E component, while an empty F component makes the pair redundant (and dually for Streett).

A run r is accepting if $\inf(r)$ satisfies α. 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 by $\mathrm{L}(\mathcal{A})$, is the set of all words in Σ^ω that $\mathcal{A}$ accepts. A nondeterministic automaton $\mathcal{A}$ is *unambiguous* if for every word $w \in \mathrm{L}(\mathcal{A})$, there is a single accepting run of $\mathcal{A}$ on w. Thus, while $\mathcal{A}$ is nondeterministic and may have many runs on each input word, it has only a single accepting run on words in its language.

We denote the different automata types by three-letter acronyms in the set $\{\mathrm{D},\mathrm{N}\}\times\{\mathrm{B},\mathrm{C},\mathrm{P},\mathrm{R},\mathrm{S}\}\times\{\mathrm{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance-condition type (Büchi, co-Büchi, parity, Rabin, or Streett); and the third indicates that we consider automata on words. For Rabin and Streett automata, we sometimes also indicate the index of the automaton. In this way, for example, NBW are nondeterministic Büchi word automata, and DRW[1] are deterministic Rabin automata with index 1.

For two automata $\mathcal{A}$ and $\mathcal{A}'$, we say that $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}')$. For an automaton type β (e.g., DBW) and an automaton $\mathcal{A}$, we say that $\mathcal{A}$ is β-realizable if there is a β-automaton equivalent to $\mathcal{A}$.

Let $\mathcal{A} = \langle \mathcal{A}, Q, Q_0, \delta, \alpha\rangle$ be an automaton. For an acceptance-condition class γ (e.g., Büchi), we say that $\mathcal{A}$ is *γ-type* if $\mathcal{A}$ has an equivalent γ automaton with the same structure as $\mathcal{A}$ [5]. That is, there is an automaton $\mathcal{A}' = \langle \Sigma, Q, Q_0, \delta, \alpha'\rangle$ such that α' is an acceptance condition of the class γ and $\mathrm{L}(\mathcal{A}') = \mathrm{L}(\mathcal{A})$.

2.2 Good-For-Games Automata

An automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha\rangle$ is *good for games* (GFG, for short) if there is a strategy $g\colon \Sigma^* \to Q$, such that for every word $w = a_1 \cdot a_2 \cdots \in \Sigma^\omega$, the sequence $g(w) = g(\epsilon), g(a_1), g(a_1 \cdot a_2), \ldots$ is a run of $\mathcal{A}$ on w, and whenever $w \in \mathrm{L}(\mathcal{A})$, then $g(w)$ is accepting. We then say that g *witnesses* $\mathcal{A}$'s GFGness.

It is known [2] that if $\mathcal{A}$ is GFG, then its GFGness can be witnessed by a finite-state strategy, thus one in which for every state $q \in Q$, the set of words $g^{-1}(q)$ is regular. Finite-state strategies can be modeled by *transducers*. Given sets I and O of input and output letters, an *(I/O)-transducer* is a tuple $\mathcal{T} = \langle I, O, M, m_0, \rho, \tau\rangle$, where M is a finite set of states, to which we refer as *memories*, $m_0 \in M$ is an *initial memory*, $\rho\colon M \times I \to M$ is a deterministic transition function, to which we refer as the *memory update function*, and $\tau\colon M \to O$ is an output function that assigns a letter in O to each memory. The transducer $\mathcal{T}$ generates a strategy $g_{\mathcal{T}}\colon I^* \to O$, obtained by following ρ and τ in the expected way: we first extend ρ to words in I^* by setting $\rho(\epsilon) = m_0$ and $\rho(u{\cdot}a) = \rho(\rho(u), a)$, and then define $g_{\mathcal{T}}(u) = \tau(\rho(u))$.

Consider a GFG automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha\rangle$, and let $g = \langle \Sigma,\ Q,\ M,\ m_0,\ \rho,\ \tau\rangle$ be a finite-state (Σ/Q)-transducer that generates a strategy $g\colon \Sigma^* \to Q$ that witnesses $\mathcal{A}$'s GFGness (we abuse notations and use g to denote both the transducer and the strategy it generates). Consider a state $q \in Q$. When $\tau(m) = q$, we say that m is a *memory of* q. We denote by $\mathcal{A}_g$ the (deterministic) automaton that models the operation of $\mathcal{A}$ when it follows g. Thus, $\mathcal{A}_g = \langle \Sigma, M, m_0, \rho, \alpha_g\rangle$, where the acceptance condition α_g is obtained from α by replacing each set $F \subseteq Q$ that appears in α (e.g. accepting states, rejecting states, set in a Rabin or Streett pair, etc) by the set $F_g = \{m \mid \tau(m) \in F\}$. Thus, $F_g \subseteq M$ contains the memories of F's states. For a state q of $\mathcal{A}$, a path π of $\mathcal{A}_g$ is *q-exclusive accepting* if π is accepting, and $\inf(\pi) \setminus \{m \mid m \text{ is a memory of } q\}$ is not accepting.

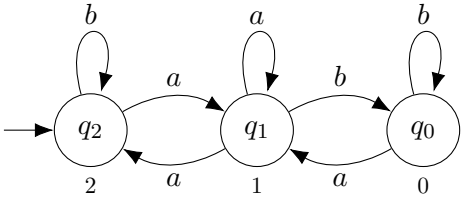

Figure 1: A weakly tight GFG-NPW $\mathcal{A}_0$. The numbers below the states describe their priorities.

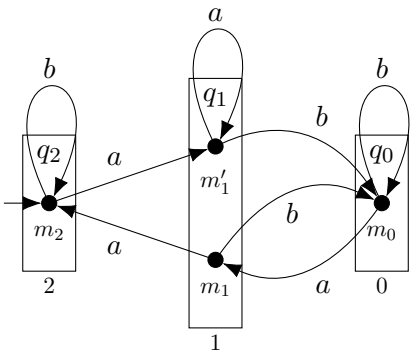

Figure 2: A strategy witnessing the GFGness of the automaton $\mathcal{A}_0$, depicted in Figure 1.

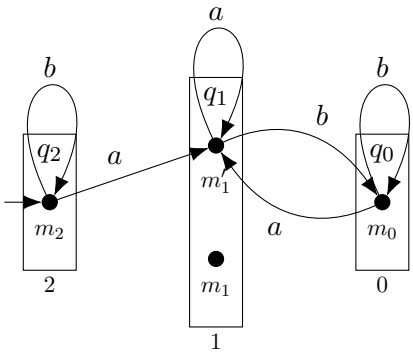

Figure 3: A strategy witnessing the tightness of a sub-automaton of $\mathcal{A}_0$.

Example 2.1. *Consider the NPW $\mathcal{A}_0$ appearing in Figure 1. We claim that $\mathcal{A}_0$ is a GFG-NPW that recognizes the language*

$$L_0 = \{w \in \{a,b\}^\omega \mid \textit{there are infinitely many } b\textit{'s in } w\}.$$

Indeed, if a word w contains only finitely many b's then $\mathcal{A}_0$ rejects w, as in all the runs of $\mathcal{A}_0$ on w, the lowest priority appearing infinitely often is 1. Therefore, $\mathrm{L}(\mathcal{A}_0) \subseteq L_0$.

We turn to describe a strategy $g\colon \{a,b\}^* \to Q$ with which $\mathcal{A}_0$ accepts all words in L_0. The only nondeterminism in $\mathcal{A}_0$ is when reading the letter a in the state q_1. Thus, we have to describe g only for words that reach q_1 and continue with an a. In that case, the strategy g moves to the state q_2, if the previous state is q_0, and to the state q_1, otherwise. Figure 2 describes a (Σ/Q)-transducer that generates g. The rectangles denote the states of $\mathcal{A}_0$, while the dots are their g-memories. The numbers below the rectangles describe the priorities of the respective states of $\mathcal{A}_0$.

As $L(\mathcal{A}_{0g}) \subseteq L(\mathcal{A}_0)$, it remains to formally prove that $L_0 \subseteq L(\mathcal{A}_{0g})$. Consider a word $w \in L_0$. Let $r = r_0, r_1, \ldots$ be the sequence of states of $\mathcal{A}_0$ visited by $\mathcal{A}_{0g}$ on w. Assume by way of contradiction that r is not accepting. Thus, r visits q_1 infinitely many times but visits q_0 only finitely many times. Let N be such that $r_m \neq q_0$ for all $m \geq N$. Consider a position $k > N$ such that $r_k = q_1$.

Since w contains infinitely many b's, there is some minimal $k' \geq k$ such that the k'-th letter in w is b. Then, $r_k = r_{k+1} = \ldots = r_{k'} = q_1$ and $r_{k'+1} = q_0$, which contradicts the choice of N. □

The following lemma generalizes known residual properties of GFG automata (c.f., [6]).

Lemma 2.2. *Consider a GFG automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ and let $g = \langle \Sigma, Q, M, m_0, \rho, \tau \rangle$ be a strategy witnessing its GFGness.*

(1) *For every state $q \in Q$ and memory $m \in M$ of q that is reachable in $\mathcal{A}_g$, we have that $\mathrm{L}(\mathcal{A}_g^m) = \mathrm{L}(\mathcal{A}^q)$.*

(2) *For every memories $m, m' \in M$ that are reachable in $\mathcal{A}_g$ with $\tau(m) = \tau(m')$, we have that $\mathrm{L}(\mathcal{A}_g^m) = \mathrm{L}(\mathcal{A}_g^{m'})$.*

Proof. We start with the first claim. Obviously, $\mathrm{L}(\mathcal{A}_g^m) \subseteq \mathrm{L}(\mathcal{A}^q)$. For the other direction, consider toward contradiction that there is a word $w \in \mathrm{L}(\mathcal{A}^q) \setminus \mathrm{L}(\mathcal{A}_g^m)$. Let u be a finite word such that $\mathcal{A}_g(u) = m$. Then, $u \cdot w \notin \mathrm{L}(\mathcal{A}_g)$. However, there is an accepting run of $\mathcal{A}$ on $u \cdot w$: it follows the run of $\mathcal{A}_g$ on u, and continues with the accepting run of $\mathcal{A}^q$ on w. Hence, g does not witness $\mathcal{A}$'s GFGness, and we have reached a contradiction. The second claim is a direct corollary of the first, as $\mathrm{L}(\mathcal{A}_g^m) = \mathrm{L}(\mathcal{A}^{\tau(m)}) = \mathrm{L}(\mathcal{A}^{\tau(m')}) = \mathrm{L}(\mathcal{A}_g^{m'})$. □

A finite *path* $\pi = q_0, \ldots, q_k$ in $\mathcal{A}$ is a sequence of states such that for $i = 0, \ldots, k-1$ we have $q_{i+1} \in \delta(q_i, a_i)$ for some $a_i \in \Sigma$. A path is a *cycle* if $q_0 = q_k$. Each path π induces a set $\mathit{states}(\pi) = \{q_0, \ldots, q_k\}$ of states in Q. A set S of finite paths then induces the set $\mathit{states}(S) = \bigcup_{\pi \in S} \mathit{states}(\pi)$. For a set P of finite paths, a *combination of paths from P* is a set $\mathit{states}(S)$ for some nonempty $S \subseteq P$.

Consider a strategy $g = \langle \Sigma, Q, M, m_0, \rho, \tau \rangle$. We say that a transition $\langle q, a, q' \rangle$ of $\mathcal{A}$ is *used by g* if there is a word $u \in \Sigma^*$ and a letter $a \in \Sigma$ such that $q = g(u)$ and $q' = g(u \cdot a)$. Consider two memories $m \neq m' \in M$ with $\tau(m) = \tau(m')$. Let $P_{m' \to m}$ be the set of paths of $\mathcal{A}_g$ from m' to m. We say that m is *replaceable by m'* if $P_{m' \to m}$ is empty or all combinations of paths from $P_{m' \to m}$ are accepting.

We say that $\mathcal{A}$ is *tight with respect to g* if all the transitions of $\mathcal{A}$ are used in g, and for all memories $m \neq m' \in M$ with $\tau(m) = \tau(m')$, we have that m is not replaceable by m'. Intuitively, the latter condition implies that both m and m' are required in g, as an attempt to merge them strictly reduces the language of $\mathcal{A}_g$. When only the first condition holds, namely when all the transitions of $\mathcal{A}$ are used in g, we say that $\mathcal{A}$ is *weakly tight* with respect to g. When a Rabin automaton $\mathcal{A}$ is *tight with respect to g*, and in addition for every state q that appears in some good set of $\mathcal{A}$'s acceptance condition, there is a q-exclusive accepting cycle in $\mathcal{A}_g$, we say that $\mathcal{A}$ is *strongly tight* with respect to g. Then, $\mathcal{A}$ is (weakly, strongly) *tight* if it is (weakly, strongly) tight with respect to some strategy.

Example 2.3. *The GFG-NPW $\mathcal{A}_0$ from Example 2.1 is weakly tight and is not tight with respect to the strategy g. Indeed, while all the transitions in $\mathcal{A}_0$ are*

used in g, the memory m_1 is replaceable by m'_1, as all combinations of paths from m'_1 to m_1 are accepting. □

The following lemma formalizes the intuition that every GFG automaton can indeed be restricted to its tight part, by removing redundant transitions and memories. Further, every tight Rabin GFG automaton has an equivalent strongly tight automaton over the same structure.

Lemma 2.4. *For every GFG automaton $\mathcal{A}$ there exists an equivalent tight GFG automaton $\mathcal{A}'$. Moreover, $\mathcal{A}'$ is defined on a substructure of $\mathcal{A}$.*

Proof. Consider a GFG automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, and let $g = \langle \Sigma, Q, M, m_0, \rho, \tau \rangle$ be a strategy that witnesses $\mathcal{A}$'s GFGness. We show how to make $\mathcal{A}$ tight with respect to a strategy obtained by merging memories in g.

As long as $\mathcal{A}$ is not tight with respect to g, we proceed as follows. First, we remove from $\mathcal{A}$ all the transitions that are not used by g. Then, if there are two memories $m, m' \in M$ with $\tau(m) = \tau(m')$ such that m is replaceable by m', we remove m from g and redirect transitions to m into m'. Note that the removal of m may cause the obtained strategy not to use some transitions in $\mathcal{A}$. We thus keep repeating both steps as long as the obtained automaton is not tight with respect to the obtained strategy.

We prove that both steps do not change the language of $\mathcal{A}$ and its GFGness. First, clearly, removal of transitions that are not used does not change the language of $\mathcal{A}$. Now, consider memories $m \neq m' \in M$ with $\tau(m) = \tau(m')$ such that m is replaceable by m'. Thus, $P_{m' \to m}$ is empty or all subsets $S \subseteq P_{m' \to m}$ are such that $\mathit{states}(S)$ is accepting. Let g' be the strategy obtained by removing m from g and redirecting transitions to m into m'.

Since $\mathrm{L}(\mathcal{A}_{g'}) \subseteq \mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}_g)$ it is enough to prove that $\mathrm{L}(\mathcal{A}_g) \subseteq \mathrm{L}(\mathcal{A}_{g'})$.

We start with the case $P_{m' \to m}$ is empty, thus there is no path from m' to m. Consider the accepting run r of $\mathcal{A}_g$ on some word w. If r does not include m, then the run of $\mathcal{A}_{g'}$ on w is identical to r, and is thus accepting. Otherwise, let p be the first position of m in r, and let w^{p+1} be the suffix of w from the position $p+1$ onwards. Since r is accepting, $w^{p+1} \in \mathrm{L}(\mathcal{A}_g^m)$. Thus, by Lemma 2.2, we have $w^{p+1} \in \mathrm{L}(\mathcal{A}_g^{m'})$. Now, since $P_{m' \to m}$ is empty, the runs of $\mathcal{A}_g^{m'}$ and $\mathcal{A}_{g'}^{m'}$ are identical on w^{p+1}, and are thus accepting. Hence, $\mathcal{A}_{g'}$ accepts w.

We continue with the case that all subsets $S \subseteq P_{m' \to m}$ are such that $\mathit{states}(S)$ is accepting. Consider a word $w \in \mathcal{A}_g$, and let r' be the run of $\mathcal{A}_{g'}$ on w. The run r' may use the memory m' instead of m finitely or infinitely many times. Consider first the case that r' uses the memory m' instead of m for k times. It is easy to prove, by an induction on k, that r' is accepting. Indeed, the base case is similar to the case $P_{m' \to m}$ is empty, and the induction step changes only a finite prefix of the run. Consider now the case that the change is done infinitely many times, in positions $p_1, p_2, \ldots$ of r'. Every path from p_i to p_{i+1} is a path from m' to m in $\mathcal{A}_g$. Hence, the set of states $\inf(r')$ is $\mathit{states}(S)$ for some nonempty $S \subseteq P_{m' \to m}$, and is thus accepting. □

Lemma 2.5. *For every tight Rabin GFG automaton, there exists an equivalent strongly tight Rabin GFG automaton over the same structure.*

Proof. Consider a tight GFG Rabin automaton $\mathcal{A}$ and let g be a strategy that witnesses $\mathcal{A}$'s GFGness and with respect to which $\mathcal{A}$ is tight. We show that the removal of redundant states from the good sets of A's accepting condition results in an automaton that is equivalent to $\mathcal{A}$ and strongly tight with respect to g.

Consider a state q of $\mathcal{A}$ that appears in some good set G of $\mathcal{A}$'s acceptance condition, and for which there is no q-exclusive accepting cycle in $\mathcal{A}_g$. We claim that the automaton $\mathcal{A}'$ that is identical to $\mathcal{A}$, except for removing q from G, is a GFG Rabin automaton equivalent to $\mathcal{A}$ that is tight w.r.t. g. Indeed:

- Regarding the language equivalence, obviously, $L(\mathcal{A}') \subseteq L(\mathcal{A})$. As for the other direction, let r be the accepting run of $\mathcal{A}_g$ on some word w. Observe that r is also an accepting run of $\mathcal{A}'_g$ on w: If q does not appear infinitely often in r then clearly r is also accepting w.r.t. $\mathcal{A}'$. Now, if q does appear infinitely often in r, then since there is no q-exclusive accepting cycle in $\mathcal{A}_g$, every cycle from q back to q is accepting w.r.t. $\mathcal{A}'$ and thus r is accepting w.r.t. $\mathcal{A}'$.

- Regarding the GFGness of $\mathcal{A}'$, since $L(\mathcal{A}) = L(\mathcal{A}_g) = L(\mathcal{A}'_g) \subseteq L(\mathcal{A}') \subseteq L(\mathcal{A})$, we get that g witnesses the GFGness of $\mathcal{A}'$.

- Regarding the tightness of $\mathcal{A}'$ w.r.t. g, observe that $\mathcal{A}$ and $\mathcal{A}'$ have the same transitions, and since $\mathcal{A}_g$ has no redundant memories, neither does $\mathcal{A}'_g$ have ones: Recall that a memory m is redundant if exists a memory m' of the same state, such that the set of paths of $\mathcal{A}_g$ from m' to m, which we denote by $P_{m' \to m}$, is empty or all combinations of paths from $P_{m' \to m}$ are accepting. The set of paths of $\mathcal{A}_g$ and of $\mathcal{A}'_g$ from m' to m are the same, and a path of $\mathcal{A}'_g$ cannot be accepting if it is not accepting in $\mathcal{A}_g$.

As there are finitely many states in $\mathcal{A}$, an iterative removal of states q as described above results in an automaton that is strongly tight w.r.t. g. □

Example 2.6. *In Figure 3 we describe a strategy g' that witnesses the tightness of a GFG-NPW on a substructure of the GFG-NPW $\mathcal{A}$ from Example 2.1. The strategy g' is obtained from g by following the procedure described in the proof of Lemma 2.4: all the transitions to m_1 are redirected to m'_1. This causes the transition (q_1, a, q_2) that was used by the memory m_1 not to be used, and it is removed.* □

A special case of GFG automata are those who are *determinizable by pruning* (or shortly DetByP) — there exists a state $q_0 \in Q_0$ and a function $\delta' \colon Q \times \Sigma \to Q$ that for every state q and letter a satisfies $\delta'(q, a) \in \delta(q, a)$ such that $\mathcal{A}' = \langle \Sigma, Q, q_0, \delta', \alpha \rangle$ is a deterministic automaton recognizing the language L($\mathcal{A}$).

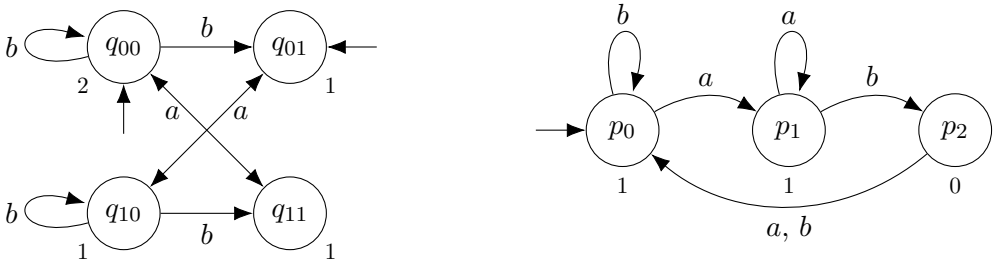

Figure 4: $\mathcal{A}_1$: An unambiguous NPW that is DBW-realizable yet is not Büchi-type.

3 Typeness Does Not Hold for Unambiguous Automata

As noted in [7], it is easy to see that typeness does not hold for nondeterministic automata: there exists an NRW that recognizes an NBW-realizable language, yet does not have an equivalent NBW on the same structure. Indeed, since all ω-regular languages are NBW-realizable, typeness in the nondeterministic setting would imply a translation of all NRWs to NBWs on the same structure, and we know that such a translation may involve a blow-up linear in the index of the NRW [17]. Even for Streett and co-Büchi automata, where the restriction to NCW-realizable languages amounts to a restriction to DCW-realizable languages, typeness does not hold.

In this section we strengthen the relation between typeness and determinism and show that typeness does not hold for nondeterministic automata even when they recognize a DBW-realizable language and, moreover, when they are unambiguous. Also, we prove the non-typeness results for NPWs, thus they apply to both Rabin and Streett automata.

Proposition 3.1. *Unambiguous NPWs are not Büchi-type with respect to DBW-realizable languages.*

Proof. Consider the automaton $\mathcal{A}_1$ depicted in Figure 4. We will show that $\mathcal{A}_1$ is unambiguous and recognizes a DBW-realizable language, yet $\mathcal{A}_1$ is not Büchi-type. Moreover, we cannot prune transitions from $\mathcal{A}_1$ and obtain an equivalent Büchi-type NPW.

The NPW $\mathcal{A}_1$ has two components: the left component, consisting of the states q_{ij}; and the right component, consisting of the states p_0, p_1, and p_2. The right part is deterministic, and it recognizes the language

$$L_{1,a,b} = \{w \in \{a,b\}^\omega \mid \text{there are infinitely many } a\text{'s and } b\text{'s in } w\}.$$

We first prove that the left component is unambiguous and that its language is:

$$L_{1,\sharp a,b} = \{w \in \{a,b\}^\omega \mid \text{there is a finite and even number of } a\text{'s in } w\ \}.$$

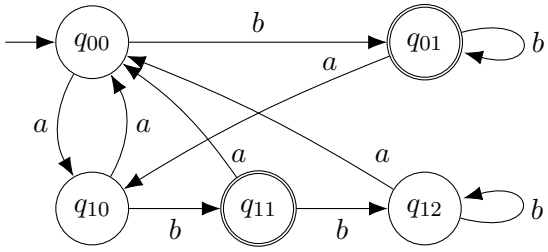

Figure 5: A DBW recognizing L_1.

To see this, observe that after reading a finite word, the left component of $\mathcal{A}_1$ can reach a state of the form q_{ij} iff $i \equiv \sharp_a(w) \pmod 2$ (i.e. i is the parity of the number of letters a in w). The only accepting runs of the left component are those that get stuck in the state q_{00}. This implies that if w is accepted by the left component, then $w \in L_{1,\sharp a,b}$. For the other direction, consider a word $w \in L_{1,\sharp a,b}$. We show that $\mathcal{A}_1$ has an (in fact, unique) accepting run on w. We can construct an accepting run of the left component of $\mathcal{A}_1$ on w by guessing whether the next block of a (i.e., a sub-word of the form a^+) has an even or odd length. If the guess is incorrect, the run is stuck reading b in a state of the form q_{i1}. If the guess is correct, the run reads the first b after the block in a state of the form q_{i0}. Thus, after reading the last block of a's, the constructed run reaches the state q_{00}, stays there forever, and $\mathcal{A}_1$ accepts w in its left component. Further, all other runs that attempt to accept w in the left component are doomed to get stuck. Thus, the left component is unambiguous.

Since $L_{1,a,b} \cap L_{1,\sharp a,b} = \emptyset$, the NPW $\mathcal{A}_1$ is unambiguous and its language is

$$L_1 = \{w \in \{a,b\}^\omega \mid w \text{ has an infinite number of } b\text{'s and an infinite or even number of } a\text{'s}\}.$$

It is not hard to see that L_1 is DBW-realizable. An example of a DBW that recognizes L_1 is depicted in Figure 5.

We prove that $\mathcal{A}_1$ is not Büchi-type. Assume by way of contradiction that there exists a subset α of $\mathcal{A}_1$'s states such that the automaton obtained form $\mathcal{A}_1$ by viewing it as an NBW with the acceptance condition α recognizes L_1. If $\{q_{00}, q_{11}\} \cap \alpha \neq \emptyset$, then the NBW accepts the word a^ω, which is not in L_1. If $\{q_{01}, q_{10}\} \cap \alpha \neq \emptyset$, then the NBW accepts the word ba^ω, which is also not in L_1. Therefore, $\alpha \subseteq \{p_0, p_1, p_2\}$. Clearly, $p_1 \notin \alpha$, as otherwise the NBW accepts a^ω. Similarly, if $p_0 \in \alpha$, then the NBW accepts ab^ω, which is also not in L_1. Thus, $\alpha = \{p_2\}$ and the NBW rejects b^ω, which is in L_1.

Finally, as $\mathcal{A}_1$ is unambiguous and all its transitions are used in the accepting run of some word, it cannot be pruned to an equivalent NPW. □

The dual case of unambiguous NPWs that are not co-Büchi-type with respect to DCW-realizable languages follows from the results of [7], and we give it here for completeness, adding the observation that the automaton described there cannot be pruned to an equivalent co-Büchi-type NPW.

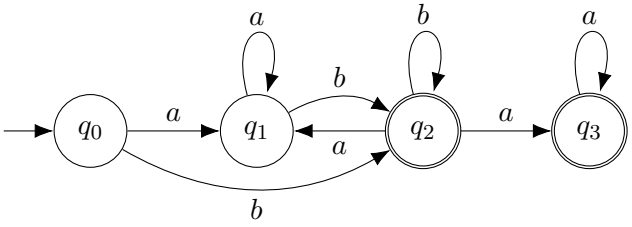

Figure 6: $\mathcal{A}_2$: An unambiguous NBW that is DCW-realizable yet is not co-Büchi-type.

Proposition 3.2. [7] *Unambiguous NPWs (and even NBWs) are not co-Büchi-type with respect to DCW-realizable languages.*

Proof. Consider the NBW $\mathcal{A}_2$ depicted in Figure 6. We will show that $\mathcal{A}_2$ is unambiguous, and recognizes a DCW-realizable language, yet $\mathcal{A}_2$ is not co-Büchi-type. Moreover, we cannot prune transitions from $\mathcal{A}_2$ for obtaining an equivalent co-Büchi-type NPW.

Notice that $\mathrm{L}(\mathcal{A}_2) = \{w \in \{a,b\}^\omega \mid w \text{ contains a letter } b\}$, which is DCW-realizable.

Yet, there is no way to define a co-Büchi acceptance condition on top of $\mathcal{A}_2$ and obtain an equivalent NCW. Moreover, as $\mathcal{A}_2$ is unambiguous and all its transitions are used in an accepting run of some word, it cannot be pruned to an equivalent one. □

We conclude this section with the following rather simple proposition, showing that automata that are both unambiguous and GFG are essentially deterministic. Essentially, it follows from the fact that by restricting an unambiguous GFG automaton $\mathcal{A}$ to reachable and nonempty states, we obtain, by pruning, a deterministic automaton, which is clearly equivalent to $\mathcal{A}$.

Proposition 3.3. *Unambiguous GFG automata are DetByP.*

Proof. Let $\mathcal{A}$ be an unambiguous GFG automaton, witnessed by a strategy g that starts in a state q_0. Without loss of generality, we can assume that $\mathrm{L}(\mathcal{A}) \neq \emptyset$. Let $\mathcal{A}'$ be the restriction of $\mathcal{A}$ to reachable and nonempty states (namely to reachable states q, such that $\mathrm{L}(\mathcal{A}^q) \neq \emptyset$). It is clear that $\mathcal{A}'$ is obtained from $\mathcal{A}$ by pruning and that $\mathrm{L}(\mathcal{A}') = \mathrm{L}(\mathcal{A})$.

We prove that $\mathcal{A}'$ is deterministic. Note first that there is a single nonempty initial state. Indeed, assume toward contradiction that there is an initial state $q_0' \neq q_0$, from which $\mathcal{A}$ has a run accepting some word w. Since $\mathcal{A}$ has an accepting run on w starting from q_0, as witnessed by g, we get a contradiction to its unambiguity.

Next, we prove that $\mathcal{A}'$ is deterministic by showing that for every finite word u over which $\mathcal{A}$ can reach a nonempty state, we have $|\mathcal{A}'(u)| = 1$. Let q be the state that $\mathcal{A}_g$ reaches when reading u and assume toward contradiction the

existence of a state $q' \neq q$, such that $q' \in \mathcal{A}'(u)$. As q' is nonempty, $\mathcal{A}^{q'}$ accepts some word w. However, since $uw \in \mathrm{L}(\mathcal{A})$, we have by the GFGness of $\mathcal{A}$ that $\mathcal{A}^q$ also accepts w. Hence, $\mathcal{A}$ has two different accepting runs on uw, contradicting its unambiguity. □

4 Co-Büchi Typeness for GFG-NSWs

In this section we study typeness for GFG-NSWs and show that, as is the case with deterministic automata, tight GFG-NSWs are co-Büchi-type. On a more technical level, the proof of Theorem 4.1 only requires the GFG automata to be weakly tight (rather than fully tight), implying that Theorem 4.1 can be strengthened in accordance. This fact is considered in Section 5, where the typeness of GFG-NRWs is shown to require full tightness.

Theorem 4.1. *Tight GFG-NSWs are co-Büchi-type: Every tight GFG-NSW that recognizes a GFG-NCW-realizable language has an equivalent GFG-NCW on the same structure.*

Proof. Consider a GFG-NSW $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, with $\alpha = \{\langle E_1, F_1 \rangle, \ldots, \langle E_k, F_k \rangle\}$. For $1 \leq i \leq k$, we refer to the sets E_i and F_i as the *bad* and *good* sets of α, respectively. Let $g = \langle \Sigma, Q, M, m_0, \rho, \tau \rangle$ be a strategy that witnesses $\mathcal{A}$'s GFGness and such that $\mathcal{A}$ is tight with respect to g. Formally, the automaton $\mathcal{A}'$ is defined as $\mathcal{A}$ with the co-Büchi acceptance condition

$$\alpha' \stackrel{\text{def}}{=} \{q \mid \text{all the cycles in } \mathcal{A}_g \text{ that go through a } g\text{-memory of } q \text{ are rejecting}\}.$$

We prove that $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}')$ and that $\mathcal{A}'$ is a GFG-NCW.

Let $Q = \{q_1, \ldots, q_n\}$. We define a sequence of NSWs $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$ and prove that: $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}_0) = \mathrm{L}(\mathcal{A}_1) = \cdots = \mathrm{L}(\mathcal{A}_n)$; g witnesses the GFGness of $\mathcal{A}_l$ for all $0 \leq l \leq n$; and $\mathcal{A}_n$ is essentially the NCW $\mathcal{A}'$. For all $0 \leq l \leq n$, the NSW $\mathcal{A}_l$ has the same structure as $\mathcal{A}$. The acceptance condition of $\mathcal{A}_l$ is $\alpha_l \cup \{\langle \alpha'_l, \emptyset \rangle\}$, where α_l and α'_l are defined as follows.

First, $\alpha_0 = \alpha$ and $\alpha'_0 = \emptyset$. Thus, going form $\mathcal{A}$ to $\mathcal{A}_0$ we only add to α a redundant pair $\langle \emptyset, \emptyset \rangle$. Clearly, $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}_0)$ and $\mathcal{A}_0$ is GFG witnessed by g.

For $1 \leq l \leq n$, we obtain α_l and α'_l from α_{l-1} and α'_{l-1} in the following way. First, we remove q_l from all the bad sets in α_{l-1}. Then, if $q_l \in \alpha'$, we add it to α'_l.

We now prove that $\mathrm{L}(\mathcal{A}_l) = \mathrm{L}(\mathcal{A}_{l-1})$ and that $\mathcal{A}_l$ is GFG witnessed by g.

We distinguish between two cases. If $q_l \in \alpha'$, the proof is not hard: adding q_l to α'_l forces it to be visited only finitely often regardless of visits in the good sets. Thus, $\mathrm{L}(\mathcal{A}_l) \subseteq \mathrm{L}(\mathcal{A}_{l-1})$. In addition, $\mathrm{L}(\mathcal{A}_{l-1}) \subseteq \mathrm{L}(\mathcal{A}_l)$, and g witnesses also the GFGness of $\mathcal{A}_l$. Indeed, an accepting run in $\mathrm{L}(\mathcal{A}_{l-1})$ remains accepting in $\mathrm{L}(\mathcal{A}_l)$. To see this, assume by way of contradiction that there is a run r that satisfies $\alpha_{l-1} \cup \{\langle \alpha'_{l-1}, \emptyset \rangle\}$ yet does not satisfy $\alpha_l \cup \{\langle \alpha'_l, \emptyset \rangle\}$. Since α_l is easier to satisfy than α_{l-1}, it must be that r violates the pair $\langle \alpha'_l, \emptyset \rangle$. Since r satisfies

the pair $\langle \alpha'_{l-1}, \emptyset \rangle$, it must visit q_l infinitely often. Since, however, $q_l \in \alpha'$, the latter indicates that r eventually traverses only rejecting cycles in $\mathcal{A}_g$ and is thus rejecting also in $\mathcal{A}_l$.

If $q_l \notin \alpha'$, we proceed as follows. Consider a state q that has a memory with an accepting cycle, and let $\mathcal{A}'$ be the NSW that is derived from $\mathcal{A}$ by taking q out of the bad sets. The change can obviously only enlarge the automaton's language. Assume toward contradiction that there is a word $w \in \mathrm{L}(\mathcal{A}') \setminus \mathrm{L}(\mathcal{A})$. Since $\mathrm{L}(\mathcal{A}') \setminus \mathrm{L}(\mathcal{A})$ is an ω-regular language, we may assume that w is a lasso word, namely of the form $w = uv^\omega$.

As the only difference between $\mathcal{A}$ and $\mathcal{A}'$ is the removal of q from bad sets, it follows that an accepting run r of $\mathcal{A}'$ on w visits q infinitely often. Hence, there are positions i and j of w, such that: I) r visits q in both i and j, II) the inner position within v is the same in positions i and j, and III) the cycle C_r that r goes through between positions i and j is accepting.

Let x be the prefix of w up to position i and y the infix of w between positions i and j. Notice that $xy^\omega = uv^\omega = w$. Consider the run r' of $\mathcal{A}'$ on w that follows r up to position j, and from there on forever repeats the cycle C_r. By the above definition of i and j, the run r' is also accepting.

Notice that since $w \notin \mathrm{L}(\mathcal{A})$, it follows that C_r is rejecting for $\mathcal{A}$. As the only difference between $\mathcal{A}$ and $\mathcal{A}'$ is the removal of q from the bad sets, it follows that combining C_r with any cycle C_a that contains q and is accepting for $\mathcal{A}$, yields a cycle that is accepting for $\mathcal{A}$. Recall that q has such an accepting cycle C_a, having that $C_r \cup C_a$ is accepting.

Since NCW=DCW, there is a DCW $\mathcal{D}$ equivalent to $\mathcal{A}$. Let n be the number of states in $\mathcal{D}$. Let z be a finite word over which $\mathcal{A}_g$ makes the cycle C_a, and consider the word $e = x(y^n z^n)^\omega$. We claim that $e \in \mathrm{L}(\mathcal{A}) \setminus \mathrm{L}(\mathcal{D})$, leading to a contradiction.

As for the positive part, $e \in \mathrm{L}(\mathcal{A})$ by the run of $\mathcal{A}$ that reaches q and then follows the C_r and C_a cycles.

Next, we show that $e \notin \mathrm{L}(\mathcal{D})$. For every $i \in \mathbb{N}$, let $e_i = x(y^n z^n)^i y^n$ be a subword of e, and let $m_i = \mathcal{A}_g(e_i)$. Notice that m_i belongs to some state q_i of $\mathcal{A}$ and not necessarily to q. By [6], the fact there exists a finite word u such that $q, q_i \in \mathcal{A}(u)$, implies that $\mathrm{L}(\mathcal{A}^q) = \mathrm{L}(\mathcal{A}^{q_i})$. Thus, since $q \in \mathcal{A}(e_i)$, we have, by Lemma 2.2, that $\mathrm{L}(\mathcal{A}_g^{m_i}) = \mathrm{L}(\mathcal{A}^{q_i}) = \mathrm{L}(\mathcal{A}^q)$.

Since $y^\omega \notin \mathrm{L}(\mathcal{A}^q)$ and $\mathrm{L}(\mathcal{A}_g^{m_i}) = \mathrm{L}(\mathcal{A}^q)$, it follows that $\mathcal{A}_g$ does not accept $x(y^n z^n)^i y^\omega$. Hence, the run of $\mathcal{D}$ on e must visit a rejecting state on every period between e_i and e_{i+1}, implying that it is rejecting.

Finally, in α_n all the bad sets are empty. Also, $\alpha'_n = \alpha'$. Thus, $\mathcal{A}_n$ is really an NCW with acceptance condition α', i.e. $\mathcal{A}'$. □

The following example shows that the weak tightness requirement cannot be omitted, even when the GFG-NSW is actually a GFG-NBW.

Example 4.2. *The automaton $\mathcal{A}_3$ depicted in Figure 8 is GFG-NBW and recognizes a GFG-NCW-realizable language, yet $\mathcal{A}_3$ has no equivalent NCW on the*

same structure.

First, it is not hard to see that $\mathrm{L}(\mathcal{A}_3) = (aa)^\omega + (aa)^* b^+ aa(b+aa)^\omega \subseteq \{a,b\}^\omega$.

Notice that if we remove the transition (q_0, b, q_0) then $\mathcal{A}_3$ becomes a deterministic automaton for the same language. In particular, $\mathcal{A}_3$ is GFG. Clearly the language of $\mathcal{A}_3$ is GFG-NCW-realizable—once the transition (q_0, b, q_0) is removed, we can make p_0 the only rejecting state, and obtain an equivalent DCW.

Now assume toward contradiction that there exists an NCW $\mathcal{A}'_3$ equivalent to $\mathcal{A}_3$ over the whole structure of $\mathcal{A}_3$. Let α' be its acceptance condition. Observe that $q_0 \notin \alpha'$ as otherwise $\mathcal{A}'_3$ rejects the word a^ω. In that case $\mathcal{A}'_3$ accepts the word b^ω, leading to a contradiction. □

5 Büchi Typeness for GFG-NRWs

Studying typeness for deterministic automata, one can use the dualities between the Büchi and co-Büchi, as well as the Rabin and Streett conditions, in order to relate the Büchi-typeness of DRWs with the co-Büchi typeness of DSWs. In the nondeterministic setting, we cannot apply duality considerations, as by dualizing a nondeterministic automaton, we obtain a universal one. As we shall see in this section, our inability to use dualization considerations is not only technical. There is an inherent difference between the co-Büchi typeness of GFG-NSWs studied in Section 4, and the Büchi typeness of GFG-NRWs, which we study here. We first show that while the proof of Theorem 4.1 only requires weak tightness, Büchi typeness requires full tightness.

The following example shows that tightness is necessary already for GFG-NCW that are GFG-NBW-realizable.

Example 5.1. *The automaton $\mathcal{A}_4$ depicted in Figure 7 is a weakly tight GFG-NCW that recognizes a GFG-NBW-realizable language, yet $\mathcal{A}_4$ has no equivalent GFG-NBW on the same structure.*

First notice that the language of $\mathcal{A}_4$ is $L_4 = a^\omega + a^* b^+ a(a+b)^\omega \subseteq \{a,b\}^\omega$. Moreover, if we remove the transitions (q_0, b, q_1) and (q_1, b, q_0), then $\mathcal{A}_4$ becomes a deterministic automaton for the same language. In particular, $\mathcal{A}_4$ is GFG. Clearly, L_4 is both DBW- and DCW-realizable.

Now assume toward contradiction that there exists an NBW $\mathcal{A}'_4$ equivalent to $\mathcal{A}_4$ over the (whole) structure of $\mathcal{A}_4$. Let α be its acceptance condition. Observe that the state q_1 must belong to α, as otherwise $\mathcal{A}'_4$ rejects the word a^ω. But in that case, $\mathcal{A}'_4$ accepts the word b^ω, leading to a contradiction. □

We now proceed to our main positive result, obtaining the typeness of GFG-NRWs.

Theorem 5.2. *Tight GFG-NRWs are Büchi-type: Every tight GFG-NRW that recognizes a GFG-NBW-realizable language has an equivalent GFG-NBW on the same structure.*

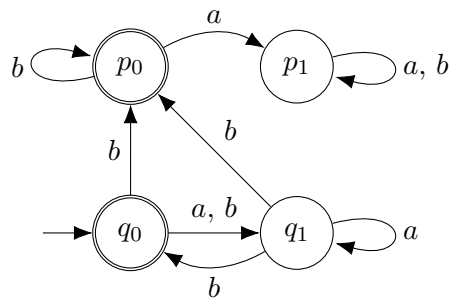

Figure 7: $\mathcal{A}_4$: A weakly tight GFG-NCW that is GFG-NBW-realizable yet is not Büchi-type.

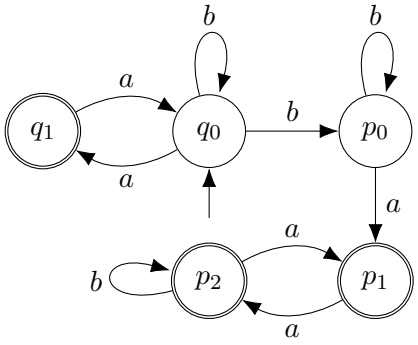

Figure 8: $\mathcal{A}_3$: A GFG-NBW that is GFG-NCW-realizable yet is not co-Büchi-type.

Consider a tight GFG-NRW $\mathcal{A}$ that recognizes a GFG-NBW-realizable language. Let g be a strategy that witnesses $\mathcal{A}$'s GFGness and with respect to which $\mathcal{A}$ is tight. By Lemma 2.5, we have a GFG Rabin automaton $\mathcal{A}'$ over the structure of $\mathcal{A}$ that is strongly tight with respect to g. We define an NBW $\mathcal{B}$ on top of $\mathcal{A}$'s structure, setting its accepting states to be all the states that appear in "good" sets of $\mathcal{A}'$ (namely in the right components of the Rabin accepting pairs).

Clearly, $\mathrm{L}(\mathcal{A}') \subseteq \mathrm{L}(\mathcal{B})$, as $\mathcal{B}$'s condition only requires the "good" part of $\mathcal{A}'$'s condition, without requiring to visit finitely often in a corresponding "bad" set. We should thus prove that $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{A}')$ and that $\mathcal{B}$ is GFG. Once proving the language equivalence, $\mathcal{B}$'s GFGness is straight forward, as the strategy g witnesses it. The language equivalence, however, is not at all straightforward.

In order to prove that $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{A}')$, we analyze the cycles of $\mathcal{A}'$ and of $\mathcal{A}'_g$, as expressed by the following lemmas.

Lemma 5.3. *Consider a GFG-NRW $\mathcal{A}$ that is GFG-NBW-realizable and a strategy g that witnesses its GFGness.*

1. *A g-memory m of a state q of $\mathcal{A}$ cannot belong to both a q-exclusive accepting cycle and a rejecting cycle.*
2. *Consider g-memories m and m' of a state q of $\mathcal{A}$, such that m belongs to a q-exclusive accepting cycle and m' belongs to a rejecting cycle. Let $P_{m\to m'}$ and $P_{m'\to m}$ be the sets of paths from m to m' and from m' to m, respectively. Then $P_{m\to m'}$ or $P_{m'\to m}$ satisfies the following property: It is empty or every combination of its paths is accepting. Formally, for $P = P_{m\to m'}$ or $P = P_{m'\to m}$, we have that $\mathit{states}(P) = \emptyset$ or $\mathit{states}(S)$ is accepting for all $S \subseteq P$.*

Proof. We start with the first claim. First, by [8], there is a DBW $\mathcal{D}$ equivalent to $\mathcal{A}$. Assume, by way of contradiction, that there are finite words p, u and v, such that $\mathcal{A}_g(p) = m$, $\mathcal{A}_g^m(u) = m$ along a q-exclusive accepting cycle, and $\mathcal{A}_g^m(v) = m$ along a rejecting cycle.

Let n be the number of states in $\mathcal{D}$, and consider the word $w = p(u^n v^n)^\omega$. For every $i \geq 1$, the NBW $\mathcal{D}$ accepts $p(u^n v^n)^i u^\omega$. Hence, it is not hard to prove that $\mathcal{D}$ also accepts w.

On the other hand, we claim that $\mathcal{A}$ does not accept w. Indeed, since v is a rejecting cycle that includes q, it must visit states in a bad set B_i for every i such that q belongs to a good set G_i. As the cycle u is q-exclusive accepting, we get that the cycle $u^n v^n$ is rejecting.

For the second claim, assume, by way of contradiction, that there are paths $\pi_1, \ldots, \pi_n \in P_{m \to m'}$ and paths $\pi'_1, \ldots, \pi'_{n'} \in P_{m' \to m}$, such that both sets of states: $\bigcup_{i=1}^{n} \mathit{states}(\pi_i)$ and $\bigcup_{i=1}^{n'} \mathit{states}(\pi'_i)$ are rejecting. Consider the path $\pi = \pi_1 \pi'_1 \pi_2 \pi'_2 \ldots \pi_n \pi'_{n'}$, where w.l.o.g. π_n is repeated until reaching the larger index n'. Then, since the union of Rabin rejecting cycles is rejecting, π is a rejecting cycle of m, contradicting the previous observation. □

Lemma 5.4. *Consider a strongly tight GFG-NRW $\mathcal{A}$ that is GFG-NBW-realizable. Then, every state q of $\mathcal{A}$ that appears in some good set has a single g-memory, and all the q-cycles in $\mathcal{A}_g$ are accepting, and at least one of them is q-exclusive.*

Proof. Since $\mathcal{A}$ is strongly tight and q appears in a good set, the "strong tightening" of $\mathcal{A}$, as per the proofs of Lemmas 2.4 and 2.5, guarantees that q has a g-memory m that belongs to a q-exclusive accepting cycle. Assume, by way of contradiction, that q has another memory $m' \neq m$. Then, due to the removal of redundant memories in Lemma 2.4, there is a rejecting combination of paths from m to m', as well as from m' to m. Hence, m belongs to a rejecting cycle, in contradiction to Lemma 5.3.

In addition, since there is a single memory m in q, and q belongs to a good set, we have, by Lemma 2.4, that m belongs to a q-exclusive accepting cycle. Hence, by Lemma 5.3, the memory m cannot also belong to a rejecting cycle. □

Lemma 5.5. *Consider a strongly tight GFG-NRW $\mathcal{A}$ that is GFG-NBW-realizable. Then, every state q of $\mathcal{A}$ that appears in some good set does not belong to a rejecting cycle.*

Proof. Assume, by way of contradiction, that q belongs to a rejecting cycle $\pi = q_0, q_1, q_2, \ldots, q_n, q_{n+1}$ with $q_0 = q_{n+1} = q$. Let S be the set of indices of bad sets that π visits. That is, an index j belongs to S if there is a state p in π that belongs to B_j. Notice that S cannot be empty, since q appears in a good set.

Let h be the maximal index of a state q_i in π up to which the strategy may exhaust the cycle states, while not adding a "fresh unrejected good state". That is:

- There is a path ρ of $\mathcal{A}_g$ from q to a memory m of q_h that visits q_i for every $1 \leq i \leq h$, and if a state p appears in ρ and in $G_i \setminus B_i$ for some acceptance set i, then $i \in S$. (Notice that the path may also visit states not in the cycle and may visit the cycle states in a different order.)
- There is no such path of $\mathcal{A}_g$ from q to q_{h+1}.

Notice that $h \geq 1$, since there is a transition $q \to q_1$ that the strategy uses, and $h \leq n$, since otherwise q belongs to a rejecting path of $\mathcal{A}_g$, while such a path does not exist due to Lemma 5.4.

Let m' be a memory of q_h that takes the transition $q_h \to q_{h+1}$. Notice that $m' \neq m$, since by the maximality of h, m does not take the transition $q_h \to q_{h+1}$.

Furthermore, there cannot be rejecting path combinations from both m to m' and from m' to m, as merging them would provide a rejecting path ρ' from m to m', which is impossible due to the maximality of h. (Concatenating ρ' to ρ provides a continuation of ρ to q_{h+1}.)

Hence, all path combinations from either m to m' or from m' to m are accepting. However, this leads to a contradiction due to the removal of redundant memories in Lemma 2.4. □

We are now in position to finish the proof of Theorem 5.2 by showing that $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{A}')$ and that $\mathcal{B}$ is GFG.

Consider a word $w \in \mathrm{L}(\mathcal{B})$, and an accepting run r of $\mathcal{B}$ on it. Let q be an accepting state that appears infinitely often in r. By Lemma 5.5, all cycles of $\mathcal{A}'$ that include q are accepting. Hence, r is also an accepting run of $\mathcal{A}'$ on w.

As for the GFGness of $\mathcal{B}$, we claim that the strategy g also witnesses $\mathcal{B}$'s GFGness. Consider a word $w \in \mathrm{L}(\mathcal{B})$. Since $\mathrm{L}(\mathcal{B}) = \mathrm{L}(\mathcal{A}') = \mathrm{L}(\mathcal{A}'_g)$, there is an accepting run r of $\mathcal{A}'_g$ on w. Therefore, there must be some state q in a good set of $\mathcal{A}'$ that is visited infinitely often along r. Thus, r is also an accepting run of $\mathcal{B}_g$ on w. This concludes the proof of Theorem 5.2. □

The following result follows directly from Lemma 2.4, Theorem 5.2, and the determinization procedure for Büchi GFG automata from [6].

Corollary 5.6. *Every GFG-NRW with n states that recognizes a DBW-realizable language has an equivalent DBW with at most $O(n^2)$ states.*

Proof. Consider a GFG-NRW $\mathcal{A}$ with n states that recognizes a DBW-realizable language. By Lemma 2.4, $\mathcal{A}$ has an equivalent tight GFG-NRW on a substructure of it, thus with at most n states. By Theorem 5.2, $\mathcal{A}$ has an equivalent GFG-NBW on the same structure, thus with at most n states. By [6], GFG-NBWs can be determinized with a quadratic blow-up, and we are done. □

6 Weak Typeness for GFG Automata

A Büchi automaton $\mathcal{A}$ is *weak* [13] if for each strongly connected component C of $\mathcal{A}$, either $C \subseteq \alpha$ (in which case we say that C is an *accepting component*) or $C \cap \alpha = \emptyset$ (in which case we say that C is a *rejecting component*). Note that a weak automaton can be viewed as both a Büchi and a co-Büchi automaton, as a run of $\mathcal{A}$ visits α infinitely often iff it gets trapped in an accepting component iff it visits states in $Q \setminus \alpha$ only finitely often. We use NWW and DWW to denote nondeterministic and deterministic weak word automata, respectively.

We show in this section that all GFG automata are type with respect to the weak acceptance condition. We provide the theorem with respect to GFG-NCWs,

from which we can easily deduce it, by our previous typeness results, also for the other types.

Theorem 6.1. *Tight GFG-NCWs are weak-type: every tight GFG-NCW that recognizes a GFG-NWW-realizable language has an equivalent GFG-NWW on the same structure.*

Proof. Consider a tight GFG-NCW $\mathcal{A}$ that recognizes a language that is GFG-NWW-realizable. Let S be the set of rejecting states of $\mathcal{A}$ and let g be a strategy witnessing $\mathcal{A}$'s tight GFGness. Let S' be the union of S and all the states q of $\mathcal{A}$ for which no g-memory m has an accepting cycle in $\mathcal{A}_g$. Let $\mathcal{A}'$ be the automaton $\mathcal{A}$ with the co-Büchi condition given by S'. Notice that the strategy g witnesses that for every state q of $\mathcal{A}'$ we have $\mathrm{L}(\mathcal{A}^q) \subseteq \mathrm{L}\big((\mathcal{A}')^q\big)$. The opposite inclusion follows from the fact that $S \subseteq S'$. Thus, $\mathcal{A}'$ is an NCW equivalent to $\mathcal{A}$ and g witnesses its GFGness.

We now prove that $\mathcal{A}'$ is weak. Assume contrarily that there exists a cycle C in $\mathcal{A}'$ that contains both a state $q \notin S'$ and a state $q' \in S'$.

Since $q \notin S'$, there is a cycle C_+ in $\mathcal{A}_g$ that is accepting in $\mathcal{A}_g$ and contains a g-memory m of q. This cycle witnesses that none of the states on C_+ can belong to $S' \setminus S$, therefore the cycle C_+ is accepting in $\mathcal{A}'_g$ as well.

We construct a cycle C_- of $\mathcal{A}'_g$ that visits some g-memory m' of q' and the g-memory m of q. This cycle is obtained by extending the cycle C of $\mathcal{A}'$ in the following way. Assume that (q_0, a_0, q_1) and (q_1, a_1, q_2) are two consecutive transitions of C. Since $\mathcal{A}'$ contains only transitions of g, these are actually transitions of $\mathcal{A}'_g$ of the form (m_0, a_0, m'_0) and (m_1, a_1, m'_1) with g-memories: m_0 of q_0; m'_0 and m_1 of q_1; and m'_1 of q_2. Notice that m'_0 may possibly be different from m_1. However, by the assumption that $\mathcal{A}$ is tight, there is a path in $\mathcal{A}'_g$ leading from m'_0 to m_1. Thus, for each pair of such consecutive transitions we can add an appropriate path to C in such a way to obtain a cycle C_- of $\mathcal{A}'_g$ that extends (as a set of states) C. Additionally, we can add to C_- two paths in such a way to visit q exactly in the g-memory m (C visits q, so it is possible as above). As $q' \in S'$ and $q' \in C \subseteq C_-$, we know that C_- is rejecting in $\mathcal{A}'_g$.

Let u_+ and u_- be the finite words over which $(\mathcal{A}')^m_g$ traverses the cycles C_+ and C_-, respectively. An infinite repetition of u_+ and u_- belongs to $\mathrm{L}\big((\mathcal{A}')^m_g\big) = \mathrm{L}\big((\mathcal{A}')^q\big) = \mathrm{L}\big(\mathcal{A}^q\big)$ if and only if it contains only finitely many copies of u_-. But this contradicts the fact that $\mathrm{L}(\mathcal{A}^q)$ can be recognized by a DWW. □

Consider now a GFG-NSW $\mathcal{A}$ that is GFG-NWW-realizable. Notice that it is obviously also GFG-NBW-realizable. Hence, by Theorem 4.1, there is a GFG-NCW on $\mathcal{A}$'s structure, and by Theorem 6.1 also a GFG-NWW. The cases of a GFG-NPW and a GFG-NBW obviously follow, since they are special cases of GFG-NSWs. As for a GFG-NRW $\mathcal{A}$ that is GFG-NWW-realizable, notice that it is obviously also GFG-NBW-realizable. Hence, by Theorem 5.2, there is a GFG-NBW on $\mathcal{A}$'s structure, and by Theorem 6.1 also a GFG-NWW.

Corollary 6.2. *Tight GFG-NSWs and GFG-NRWs are weak-type: every tight GFG-NSW and GFG-NRW that recognizes a GFG-NWW-realizable language has an equivalent GFG-NWW on the same structure.*

Next, we show that GFG-NWWs are DetByP, generalizing a folklore result about safe and co-safe GFG automata.

Theorem 6.3. *GFG-NWWs are DetByP.*

Proof. Consider a GFG-NWW $\mathcal{A}$ with accepting set α. By Lemmas 2.4 and 2.5, we may assume that $\mathcal{A}$ is strongly tight w.r.t. a strategy g. First notice that by Lemma 5.4, a state $q \in \alpha$ has only one g-memory, and is therefore already deterministic.

Now we consider the case of a state $q \notin \alpha$ such that there are at least two g-memories m and m' of q. Let g' be the strategy obtained by removing m' from g and redirecting transitions to m' into m.

We now show that $\mathrm{L}(\mathcal{A}_g) = \mathrm{L}(\mathcal{A}_{g'})$. From that, by induction it follows that the number of memories of each state of $\mathcal{A}$ can be reduced to 1.

Consider a word $w \in \mathcal{A}_g$, and let r' be the run of $\mathcal{A}_{g'}$ on w. The run r' may use the memory m instead of m' finitely or infinitely many times. If r' uses it only finitely many times, then by an argument similar to the one given in the proof of Lemma 2.4, r' is also accepting. (The argument inductively uses Lemma 2.2, according to which $\mathrm{L}(\mathcal{A}^m_g) = \mathrm{L}(\mathcal{A}^{m'}_g)$.)

We continue with the case that the change is done infinitely many times, in positions $p_1, p_2, \ldots$ of r', and assume toward contradiction that r' is rejecting. Every path from p_i to p_{i+1} is a path from m to m' in $\mathcal{A}_g$. Notice that the suffix of w from position p_1 onwards is in $\mathrm{L}(\mathcal{A}^q) \setminus \mathrm{L}(\mathcal{A}^m_{g'})$. Since we consider ω-regular languages, we can assume without loss of generality that this suffix is periodic, in the form of u^ω, where $\mathcal{A}^m_g(u) = m'$. Let u' be a finite word such that $\mathcal{A}^{m'}_g(u') = m$.

Consider now a word $w' \in (u + u')^\omega$. First assume that w' contains only finitely many instances of u'. In that case, we have that $\mathcal{A}^q$ accepts w', because $\mathcal{A}^q$ has a run that loops back to q until reading the last occurrence of u' and then follows the run witnessing that $u^\omega \in \mathrm{L}(\mathcal{A}^q)$. We refer to such words as of the *first kind*.

Now assume that a suffix of w' from some point on is equal to $(uu')^\omega$. Since $(uu')^\omega \notin \mathrm{L}(\mathcal{A}^m_g)$, we get by Lemma 2.2 that $w' \notin \mathrm{L}(\mathcal{A}^q)$. We refer to such words as of the *second kind*.

Now consider the minimal, namely last, strongly-connected component of $\mathcal{A}_g$ that can be reached from m by reading words in the language $(u + u')^*$. If this component is accepting, then $\mathcal{A}^m_g$ accepts a word of the second kind. Similarly, if the component is rejecting then $\mathcal{A}^m_g$ rejects a word of the first kind. In both cases we get a contradiction. □

By combining the above results, we obtain the following corollary.

Corollary 6.4. *Every GFG-NSW and GFG-NRW that recognizes a GFG-NWW-realizable language is DetByP.*

7 Consequences

GFG automata provide an interesting formalism in between deterministic and nondeterministic automata. Their translation to deterministic automata is immediate for the weak condition (Theorem 6.3), polynomial for the Büchi condition [6], and exponential for the co-Büchi, parity, Rabin, and Streett conditions [6]. They have the same typeness behavior as deterministic automata, summarized in Table 1. The positive results of Table 1 follow from our theorems in Sections 4, 5, and 6. The negative results follow from corresponding counterexamples with deterministic automata [5, 7]. Considering the complementation of GFG automata, they lie in between the deterministic and nondeterministic settings, as shown in Table 2. As for the translation of LTL formulas to GFG automata, it is doubly exponential, like the translation to deterministic automata (Corollary 7.3 below).

Complementation In the deterministic setting, Rabin and Streett automata are dual: complementing a DRW into a DSW, and vice versa, is simply done by switching between the two acceptance conditions on top of the same structure. This is not the case with GFG automata. We show below that complementing a GFG-NSW, and even a GFG-NCW, into a GFG-NRW involves an exponential state blow-up. Essentially, it follows from the Büchi-typeness of GFG-NRWs (Theorem 5.2) and the fact that while determinization of GFG-NBWs involves only a quadratic blow-up, determinization of GFG-NCWs involves an exponential one [6].

Corollary 7.1. *The complementation of a GFG-NCW into a GFG-NRW involves a $2^{\Omega(n)}$ state blow-up.*

Proof. By [6], there is a GFG-NCW $\mathcal{C}$ with n states whose equivalent DCWs must have at least $2^{\Omega(n)}$ states. Consider a GFG-NRW $\mathcal{A}$ with x states for the complement of $\mathcal{C}$.

Since the language of $\mathcal{A}$ is DBW-recognizable, then, by Corollary 5.6, there is a DBW $\mathcal{D}$ equivalent to $\mathcal{A}$ whose state space is quadratic in the number of states of $\mathcal{A}$, namely with up to x^2 states. As the dual of $\mathcal{D}$ is a DCW equivalent to $\mathcal{C}$, it follows that $\mathcal{D}$ has at least $2^{\Omega(n)}$ states. Hence, $x^2 \geq 2^{\Omega(n)}$, implying that $x \geq 2^{\Omega(n/2)} = 2^{\Omega(n)}$. □

Using our typeness results, we get an almost complete picture on complementation of GFG automata.

Theorem 7.2. *The state blow-up involved in the complementation of GFG automata is as summarized in Table 2.*

Proof.

- From weak and Büchi. A GFG-NBW $\mathcal{A}$ with n states has an equivalent DBW $\mathcal{D}$ with up to n^2 states [6], on which structure there is a DCW $\overline{\mathcal{D}}$ for

Type To / From	W	B	C	P	R	S
Weak	Yes					
Büchi						
Co-Büchi						
Parity						
Rabin			No		Y	N
Streett		N	Y	No		Y

Table 1: Typeness in translations between GFG automata. (Y=Yes; N=No.)

Comp. To / From	$\overline{\text{W}}$	$\overline{\text{C}}$	$\overline{\text{B}}$	$\overline{\text{P}}$	$\overline{\text{R}}$	$\overline{\text{S}}$
Weak	Poly					
Büchi						
Co-Büchi			Exp			?
Parity						
Rabin						
Streett						

Table 2: The state blow-up involved in the complementation of GFG automata.

the complement language. Notice that $\overline{\mathcal{D}}$ is also a GFG-NCW, GFG-NPW, GFG-NRW, and GFG-NSW. Now, if there is a GFG-NBW equivalent to $\overline{\mathcal{D}}$, then $\overline{\mathcal{D}}$ is DWW-recognizable, and, by Theorem 6.1, there is a GFG-NWW on a substructure of $\overline{\mathcal{D}}$.

- From co-Büchi. By Corollary 7.1, we have the exponential state blow-up in the complementation to GFG-NPW and GFG-NRW automata. Since the complement of a co-Büchi-recognizable language is DBW-recognizable, we get an exponential state blow-up also to GFG-NBW.

- To weak and co-Büchi. Consider a GFG-NCW, GFG-NPW, or GFG-NRW $\mathcal{A}$ with n states that can be complemented into a GFG-NCW $\mathcal{C}$. Then the language of $\mathcal{A}$ is GFG-NBW recognizable. Thus, by Theorem 5.2, there is a GFG-NBW equivalent to $\mathcal{A}$ with up to n states. Hence, by case (1), there is a GFG-NCW for the complement of $\mathcal{A}$ with up to n^2 states.

- From Streett to weak. Consider a GFG-NSW $\mathcal{A}$ that can be complemented to a GFG-NWW. Then the language of $\mathcal{A}$ is DWW-recognizable. Thus, by Theorems 4.1 and 6.1, there is a GFG-NWW on a substructure of $\mathcal{A}$, and we are back in case (1).

- From Streett to co-Büchi. Given a DRW $\mathcal{A}$ that is NCW realizable, one can translate it to an equivalent NCW by first dualizing $\mathcal{A}$ into a DSW $\overline{\mathcal{A}}$ for the complement language, and then complementing $\overline{\mathcal{A}}$ into a GFG-NCW $\mathcal{C}$. Since dualizing a DRW into a DSW is done with no state blowup and the translation of DRWs to NCWs might involve an exponential state blowup [1], so does the complementation of GFG-NSW to GFG-NCWs.

- From Streett to Streett. Analogous to the above case of Streett to co-Büchi, due to the exponential state blowup in the translation of DRWs to NSWs [1]. □

Translating LTL formulas to GFG Automata Recall that GFG-NCWs are exponentially more succinct than DCWs [6], suggesting they do have some power of nondeterministic automata. A natural question is whether one can come up with an exponential translation of LTL formulas to GFG automata, in particular when attention is restricted to LTL formulas that are DCW-realizable. We complete this section with a negative answer, providing another evidence for the deterministic nature of GFG automata. This result is based on the fact that the language with which the doubly-exponential lower bound of the translation of LTL to DBW in [9] is proven is bounded (that is, it is both safe and co-safe). It means that by Corollary 6.4, any GFG-NSW for it would be DetByP, contradicting the doubly-exponential lower bound.

Corollary 7.3. *The translation of DCW-realizable LTL formulas into GFG-NSW is doubly exponential.*

References

[1] U. Boker. Rabin vs. Streett automata. In *Proc. 37th Conf. on Foundations of Software Technology and Theoretical Computer Science*, pages 17:1–17:15, 2017.

[2] U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100, 2013.

[3] T. Colcombet and C. Löding. Regular cost functions over finite trees. In *Proc. 25th IEEE Symp. on Logic in Computer Science*, pages 70–79, 2010.

[4] 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.

[5] S.C. Krishnan, A. Puri, and R.K. Brayton. Deterministic ω-automata vis-a-vis deterministic Büchi automata. In *Algorithms and Computations*, volume 834 of *Lecture Notes in Computer Science*, pages 378–386. Springer, 1994.

[6] D. Kuperberg and M. Skrzypczak. On determinisation of Good-For-Games automata. In *Proc. 42nd Int. Colloq. on Automata, Languages, and Programming*, pages 299–310, 2015.

[7] O. Kupferman, G. Morgenstern, and A. Murano. Typeness for ω-regular automata. *International Journal on the Foundations of Computer Science*, 17(4):869–884, 2006.

[8] O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.

[9] O. Kupferman and M.Y. Vardi. From linear time to branching time. *ACM Transactions on Computational Logic*, 6(2):273–294, 2005.

[10] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.

[11] L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[12] G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.

[13] D.E. Muller, A. Saoudi, and P.E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, 1988.

[14] D. Niwiński 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.

[15] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[16] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

[17] H. Seidl and D. Niwiński. On distributive fixed-point expressions. *Theoretical Informatics and Applications*, 33(4–5):427–446, 1999.

[18] W. Thomas. On the synthesis of strategies in infinite games. In E.W. Mayr and C. Puech, editors, *Proc. 12th Symp. on Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995.

[19] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.