On (I/O)-aware Good-For-Games Automata

Rachel Faran and Orna Kupferman

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Abstract. Good-For-Games (GFG) automata are nondeterministic automata that can resolve their nondeterministic choices based on the past. The fact that the synthesis problem can be reduced to solving a game on top of a GFG automaton for the specification (that is, no determinization is needed) has made them the subject of extensive research in the last years. GFG automata are defined for general alphabets, whereas in the synthesis problem, the specification is over an alphabet $2^{I \cup O}$, for sets I and O of input and output signals, respectively. We introduce and study (I/O)-*aware GFG automata*, which distinguish between nondeterminism due to I and O: both should be resolved in a way that depends only on the past; but while nondeterminism in I is hostile, and all I-futures should be accepted, nondeterminism in O is cooperative, and a single O-future may be accepted. We show that (I/O)-aware GFG automata can be used for synthesis, study their properties, special cases and variants, and argue for their usefulness. In particular, (I/O)-aware GFG automata are unboundedly more succinct than deterministic and even GFG automata, using them circumvents determinization, and their study leads to new and interesting insights about hostile vs. collaborative nondeterminism, as well as the theoretical bound for realizing systems.

1 Introduction

Synthesis is the automated construction of systems from their specifications [6, 18]. The system should *realize* the specification, namely satisfy it against all possible environments. More formally, the specification is a language L of infinite words over an alphabet $2^{I \cup O}$, where I and O are sets of input and output signals, respectively, and the goal is to build a reactive system that outputs assignments to the signals in O upon receiving assignments to the signals in I, such that the generated sequence of assignments, which can be viewed as an infinite computation in $(2^{I \cup O})^\omega$, is in L [18]. The common approach for solving the synthesis problem is to define a two-player game on top of a deterministic automaton $\mathcal{D}$ for L. The positions of the game are the states of $\mathcal{D}$. In each round of the game, one player (the environment) provides an input assignment in 2^I, the second player (the system) responds with an output assignment in 2^O, and the game transits to the corresponding successor state. The goal of the system is to respond in a way so that the sequence of visited positions satisfies the acceptance condition of $\mathcal{D}$. Thus, the generated computation is in L. The system has a winning strategy in the game iff the language L is (I/O)-realizable [9].

Now, if one replaces $\mathcal{D}$ by a nondeterministic automaton $\mathcal{A}$ for L, the system has to respond not only with an output, but also with a transition of $\mathcal{A}$ that should be taken. This is problematic, as this choice of a transition should accommodate all possible future choices of the environment. In particular, if different future choices of the environment induce computations that are all in the language of $\mathcal{A}$ yet require different nondeterministic choices, the system cannot win. Thus, it might be that L is realizable and still the system has no winning strategy in the game.

Some nondeterministic automata are, however, good for games. The study of such automata started in [13], by means of tree automata for derived languages. It then continued by means of *good for games* (GFG) word automata [11].[1] Intuitively, a nondeterministic automaton $\mathcal{A}$ is GFG if it is possible to resolve its nondeterminism in a manner that only depends on the past and still accepts all the words in the language. Formally, $\mathcal{A}$ over an alphabet Σ and state space Q is GFG if there is a *strategy* $g : \Sigma^* \to Q$, such that for every word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$, the sequence $g(w) = g(\epsilon), g(\sigma_1), g(\sigma_1 \cdot \sigma_2),\ldots$ is a run of $\mathcal{A}$ on w, and whenever w is accepted by $\mathcal{A}$, the run $g(w)$ is accepting. Thus, the strategy g, which *witnesses* $\mathcal{A}$'s GFGness, maps each word $x \in \Sigma^*$ to the state that is visited after x is read. 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.[2] In terms of expressive power, it is shown in [13, 17] that GFG automata with an acceptance condition γ (e.g., Büchi) are as expressive as deterministic γ automata. In terms of succinctness, GFG automata on infinite words are more succinct (possibly even exponentially) than deterministic ones [4, 12]. Further research studies decidability, typeness, complementation, construction, and minimization for GFG automata [12, 5, 3, 1], as well as GFG automata for ω-pushdown languages [15]. Beyond its computational advantages, the use of GFG automata circumvents cumbersome determinization constructions that traditional synthesis algorithms involve [19, 14].

Recall that in order to be GFG, an automaton needs a strategy $g : \Sigma^* \to Q$ that resolves nondeterminism in a way that depends only on the past. We argue that this is a too strong requirement for the synthesis problem. There, $\Sigma = 2^{I \cup O}$, and we suggest to distinguish between nondeterminism due to the 2^I component of each letter, which is hostile, and nondeterminism due to the 2^O component, which is cooperative. As a simple example, consider the nondeterministic Büchi (in fact, looping) automaton $\mathcal{A}_1$ over $2^{\{a,b\}}$ appearing in Figure 1. The Boolean assertions on the transitions describe the letters with which they can be taken. For example, the transition from q_0 to q_1 can be taken with the letters $\{a\}$ or $\{a, b\}$. Note that $\mathcal{A}_1$ is not GFG. Indeed, a strategy $g : (2^{\{a,b\}})^* \to Q$ neglects either the word $\{a\}^\omega$, in the case $g(\{a\}) = q_1$, or the word $\{a\} \cdot \{a, b\}^\omega$, in the case $g(\{a\}) = q_2$. Assume that a is an input signal and b is an output signal, and that we play the synthesis game on top of $\mathcal{A}_1$. Since the system controls the assignment to b, it wins the game: on input $\{a\}$, it can proceed to q_1, and keep assigning true to b, staying forever in q_1, or it can proceed to q_2 and keep assigning false to b, staying forever in q_2.

We introduce and study (I/O)*-aware GFG automata*, which distinguish between nondeterminism due to I and O: both should be resolved in a way that depends on the past; but while nondeterminism in I is hostile, and the strategy witnessing the GFGness should address all possible "I-futures", nondeterminism in O is cooperative, and a single "O-future", which the strategy chooses, is sufficient. More formally, an automaton $\mathcal{A}$ over $2^{I \cup O}$ is (I/O)-aware GFG if for every word $w_I \in (2^I)^\omega$, if w_I is *hopeful*, namely it can be paired with a word $w_O \in (2^O)^\omega$ to a computation accepted by $\mathcal{A}$, then the pairing as well as the accepting run of $\mathcal{A}$ can be produced in an on-line manner, thus in a way that only depends

[1] GFGness is also used in [7] in the framework of cost functions under the name "history-determinism".

[2] In fact, DBP automata were the only examples known for GFG automata when the latter were introduced in [11]. As explained there, however, even DBP automata are useful in practice, as their transition relation is simpler than the one of the embodied deterministic automaton and it can be defined symbolically.

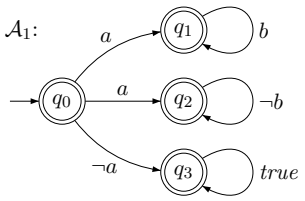

Fig. 1. The automaton $\mathcal{A}_1$ is not GFG, yet is $(\{a\}/\{b\})$-aware GFG.

on the past. For example, the automaton $\mathcal{A}_1$ from Figure 1 is $(\{a\}/\{b\})$-aware GFG, as given the a-component of a letter, there is a strategy that pairs it with a b-component and a transition of $\mathcal{A}_1$ in a way that all the hopeful words in $(2^{\{a\}})^\omega$ are paired with a word in $(2^{\{b\}})^\omega$ and an accepting run on the obtained computation.

After introducing (I/O)-aware GFG automata, our first set of results concerns their applications and decidability. First, we show that nondeterminisitic (I/O)-aware GFG automata are *sound and complete for* (I/O)*-realizability*: the system has a winning strategy in a game played on them iff the specification is (I/O)-realizable. Note that using a nondeterministic automaton is always sound. The use of deterministic automata, then GFG automata, and now (I/O)-aware GFG automata, is required for the completeness. We conclude that the synthesis problem for (I/O)-aware GFG automata with acceptance condition γ can be solved in the same complexity as deciding games with γ winning conditions. Thus, it coincides with the complexity for deterministic automata. In particular, for (I/O)-aware nondeterministic Büchi automata, the synthesis problem can be solved in quadratic time. Then, we study the problem of deciding whether a given nondeterministic automaton is (I/O)-aware GFG. We show that the problem is reducible to the problem of deciding whether the projection of $\mathcal{A}$ on I is GFG, and following [3], conclude that it is polynomial for Büchi automata. We also extend the notion of DBP automata to the (I/O)-aware setting, and study the relation between DBP and (I/O)-aware DBP automata, as well as the relation between (I/O)-aware DBP and (I/O)-aware GFG automata.

It is tempting to believe that the more signals we identify as outputs, the "more (I/O)-aware GFG" the automaton is. Our second set of results has to do with the fact that the above intuition is wrong. Essentially, this follows from the fact that while nondeterminism in O is cooperative, a strategy that witnesses (I/O)-aware GFGness has to "cover" only hopeful input words, and the identification of signals as outputs may make some input words hopeful. In particular, while all deterministic automata are GFG, not all deterministic automata are (I/O)-aware GFG. In order to address this phenomenon, we introduce (I^+/O^-)*-aware GFG automata*: an automaton $\mathcal{A}$ is (I^+/O^-)-aware GFG if there is a partition $\langle I', O'\rangle$ of $I \cup O$ such that $I \subseteq I'$ and $\mathcal{A}$ is (I'/O')-aware GFG. Intuitively, since $I \subseteq I'$, then the system has less control in the $\langle I', O'\rangle$ partition, which we show to imply that (I^+/O^-)-aware GFG automata are sound and complete for (I/O)-realizablity. As discussed above, however, the connection between controlability and GFGness is not monotone. Consequently, while deciding (I/O)-aware GFGness for Büchi automata is polynomial, we show that deciding their (I^+/O^-)-aware GFGness requires a check of all possible partitions of $I \cup O$, and is NP-complete in $|I \cup O|$.

(I/O)-aware GFG automata significantly extend the type of automata that are sound and complete for (I/O)-realizability. A natural problem that follows is the generation of small (I^+/O^-)-aware GFG automata. Our third set of results concerns this challenge, and its

tight relation to the problem of generating small realizing (I/O)-transducers. We describe two heuristics in this front. In the first, we introduce the notion of (I/O)-*coverage* between automata, which together with (I/O)-aware GFGness entails preservation of (I/O)-realizability. We then discuss generation of small (I/O)-covering automata, showing that (I/O)-aware GFG automata are unboundedly more succinct than deterministic and even GFG automata. Intuitively, while an automaton may need a large state space in order to recognize all computations, an (I/O)-aware GFG automaton that is used for synthesis may reject some of the computations, as long as it covers all hopeful input words. While (I/O)-covering GFG automata under-approximate the specification, our second heuristic is *counter-example guided inductive synthesis* (CEGIS), and it generates over-approximating GFG automata. Unlike earlier CEGIS efforts [21, 20, 2], our starting point is a LTL formula, and we iteratively refine GFG automata that over-approximate the specification and its complement. Our GFG automata are obtained by applying the subset construction on the nondeterministic automaton and adding nondeterministic transitions to states associated with strict subsets of the successor subset. Working with the subset construction is always sound and complete for safety automata. For Büchi automata, refinement steps are needed in cases richer information that is needed for keeping track of visits in accepting states. Working with GFG automata, we let the winning strategy use small subsets, in particular follow the nondeterministic automaton when possible. In Section 8, we elaborate further on how our results shed light on the open problem of whether a minimal realizing transducer for an NBW specification with n states needs $2^{O(n \log n)}$ or only $2^{O(n)}$ states, as well as the use of GFG automata whose choices under-approximate the specifications.

2 Preliminaries

2.1 Automata

A *nondeterministic word automaton* over a finite alphabet Σ is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Q is a set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to 2^Q \setminus \{\emptyset\}$ is a total transition function, and α is an acceptance condition. We say that $\mathcal{A}$ is *deterministic* if for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| = 1$. A *run* of $\mathcal{A}$ on an infinite word $\sigma_0, \sigma_1, \cdots \in \Sigma^\omega$ is a sequence of states $r = q_0, q_1, \ldots$, where for every position $i \geq 0$, we have that $q_{i+1} \in \delta(q_i, \sigma_i)$. We use *inf*$(r)$ to denote the set of states that r visits infinitely often. Thus, *inf*$(r) = \{q : \ q_i = q \text{ for infinitely many } i\text{-s}\}$.

We consider *Büchi* acceptance condition, where $\alpha \subseteq Q$, and a run is accepting iff it visits states in α infinitely often; that is, $\alpha \cap$ *inf*$(r) \neq \emptyset$. We also consider *looping* automata, which are a special case of Büchi automata in which all states except for one rejecting sink are in α (equivalently, $\alpha = Q$ and the transition function need not be total). The *language* of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all words $w \in \Sigma^\omega$ such that $\mathcal{A}$ has an accepting run on w.

We use three letter acronyms in $\{\text{D,N}\} \times \{\text{B,L}\} \times \{\text{W}\}$ to denote classes of word automata. The first letter indicates whether this is a deterministic or nondeterministic automaton, and the second indicates the acceptance condition. For example, NLW is a nondeterministic looping automaton.

We say that a nondeterministic automaton is *good for games* (GFG, for short) if its nondeterminism can be resolved based on the past [11]. Formally, a nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ is GFG if there exists a function $g : \Sigma^* \to Q$ such that the following hold: (1) $g(\epsilon) = q_0$, (2) The strategy g is compatible with δ; thus, for every

$w \cdot \sigma \in \Sigma^* \times \Sigma$, we have that $g(w \cdot \sigma) \in \delta(g(w), \sigma)$, and (3) The strategy g "covers" all words in $L(\mathcal{A})$; thus for every word $w = \sigma_0 \cdot \sigma_1 \cdots \in L(\mathcal{A})$, the run that g induces on w, namely $g(\epsilon), g(\sigma_0), g(\sigma_0 \cdot \sigma_1), \ldots$, is accepting. We then say that g *witnesses* the GFGness of $\mathcal{A}$.

2.2 Games

A *game* is a tuple $G = \langle V_{\text{AND}}, V_{\text{OR}}, E, \alpha \rangle$, where V_{AND} and V_{OR} are disjoint sets of positions, owned by Player AND and Player OR, respectively. Let $V = V_{\text{AND}} \cup V_{\text{OR}}$. Then, $E \subseteq V \times V$ is an edge relation, and α is a winning condition, defining a subset of V^ω. A *play* is an infinite sequence of positions $v_0, v_1, \cdots \in V^\omega$, such that for every index $i \geq 0$, we have that $\langle v_i, v_{i+1} \rangle \in E$. A play $\pi \in V^\omega$ is *winning* for Player OR if π satisfies α, and is winning for Player AND otherwise. We focus here on *Büchi* games, where $\alpha \subseteq V$ and π satisfies α if it visits the positions in α infinitely often.

Starting from some position $v_0 \in V$, the players generate a play in G as follows. In every round, if the current position is $v \in V_j$, for $j \in \{\text{AND}, \text{OR}\}$, then Player j chooses a successor v' of v, and the play proceeds to position v'. A *strategy* for a player $j \in \{\text{AND}, \text{OR}\}$ is a function $f_j : V^* \times V_j \to V$ such that for every $u \in V^*$ and $v \in V_j$, we have that $\langle v, f_j(u, v) \rangle \in E$. Thus, a strategy for Player j maps the history of the game so far, when it ends in a position v owned by Player j, to a successor of v. Two strategies $f_{\text{AND}}, f_{\text{OR}}$, and an initial position v_0 induce a play $\pi = v_0, v_1, v_2 \cdots \in V^\omega$, where for every $i \geq 0$, if $v_i \in V_j$, for $j \in \{\text{AND}, \text{OR}\}$, then $v_{i+1} = f_j((v_0, \ldots, v_{i-1}), v_i)$. We say that π is the *outcome* of $f_{\text{OR}}, f_{\text{AND}}$, and v_0, and denote $\pi = outcome(f_{\text{OR}}, f_{\text{AND}}, v_0)$.

We say that a position $v \in V$ is winning for Player OR if there exists a strategy f_{OR} such that for every strategy f_{AND}, we have that $outcome(f_{\text{OR}}, f_{\text{AND}}, v)$ is winning for Player OR. We then say that f_{OR} is a *winning strategy* of Player OR from v. We define similarly winning positions and strategies for Player AND.

It is known that Büchi games are *determined*. That is, every position in a Büchi game is winning for exactly one of the players. Solving a game is deciding which vertices are winning for every player. Büchi games can be solved in quadratic time [22].

2.3 Synthesis

Consider two finite sets I and O of input and output signals, respectively. For two words $w_I = i_0 \cdot i_1 \cdot i_2 \cdots \in (2^I)^\omega$ and $w_O = o_0 \cdot o_1 \cdot o_2 \cdots \in (2^O)^\omega$, we define $w_I \oplus w_O$ as the word in $(2^{I \cup O})^\omega$ obtained by merging w_I and w_O. Thus, $w_I \oplus w_O = (i_0 \cup o_0) \cdot (i_1 \cup o_1) \cdot (i_2 \cup o_2) \cdots$.

An (I/O)-*transducer* models a finite-state system that generates assignments to the output signals while interacting with an environment that generate assignments to the input signals. Formally, an (I/O)-transducer is $\mathcal{T} = \langle I, O, S, s_0, \rho, \tau \rangle$, where S is a set of states, $s_0 \in S$ is an initial state, $\rho : S \times 2^I \to S$ is a transition function, and $\tau : S \to 2^O$ is a labelling function on the states. Intuitively, $\mathcal{T}$ models the interaction of an environment that generates at each moment in time a letter in 2^I with a system that responds with letters in 2^O. Consider an input word $w_I = i_0 \cdot i_1 \cdot \cdots \in (2^I)^\omega$. The *run* of $\mathcal{T}$ on w_I is the sequence $s_0, s_1, s_2 \ldots$ such that for all $j \geq 0$, we have that $s_{j+1} = \rho(s_j, i_j)$. The *output* of $\mathcal{T}$ on w_I is then $w_O = o_1 \cdot o_2 \cdot \cdots \in (2^O)^\omega$, where $o_j = \tau(s_j)$ for all $j \geq 1$. Note that the first output assignment is that of s_1, thus $\tau(s_0)$ is ignored. This reflects the fact that the environment

initiates the interaction. The *computation of* $\mathcal{T}$ *on* w_I, denoted $\mathcal{T}(w_I)$, is then $w_I \oplus w_O$. Thus, $\mathcal{T}(w_I) = i_0 \cup o_1, i_1 \cup o_2, \ldots \in (2^{I \cup O})^\omega$.

For an automaton $\mathcal{A}$ over $2^{I \cup O}$, we say that $\mathcal{T}$ (I/O)-*realizes* $\mathcal{A}$ if for every input word $w_I \in (2^I)^\omega$, the computation of $\mathcal{T}$ on w_I is in $L(\mathcal{A})$. If there exists an (I/O)-transducer $\mathcal{T}$ that (I/O)-realizes $\mathcal{A}$, then we say that $\mathcal{A}$ is (I/O)-realizable. The *synthesis* problem is to decide, given an automaton $\mathcal{A}$, whether $\mathcal{A}$ is (I/O)-realizable, and if so, to return an (I/O)-transducer that realizes it.

A common approach to solve the synthesis problem uses games. Given an NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$ with a total δ, we define the *synthesis game* $G_{\text{SYN}}(\mathcal{A}, I, O)$ as follows. Intuitively, the game is played over Q, and starts at q_0. Let q be the position of the game at the beginning of some round. The round proceeds as follows: first, Player AND, who represents the environment, chooses a letter $i \in 2^I$. Then, Player OR, who represents the system, chooses a letter $o \in 2^O$ and a state q' such that $q' \in \delta(q, i \cup o)$, and the game proceeds to q'. Formally, we define $G_{\text{SYN}}(\mathcal{A}, I, O) = \langle V_{\text{AND}}, V_{\text{OR}}, E, \alpha \rangle$, where $V_{\text{AND}} = Q$, $V_{\text{OR}} = Q \times 2^I$, and $E = \{\langle q, \langle q, i \rangle\rangle : q \in V_{\text{AND}} \text{ and } i \in 2^I\} \cup \{\langle \langle q, i \rangle, q' \rangle : \text{ there is } o \in 2^O \text{ such that } q' \in \delta(q, i \cup o)\}$.

We say that an automaton $\mathcal{A}$ over alphabet $2^{I \cup O}$ is *sound and complete for* (I/O)-*realizability* when $\mathcal{A}$ is (I/O)-realizable iff q_0 is a winning state for Player OR in $G_{\text{SYN}}(\mathcal{A}, I, O)$. Note that all NBWs are sound for (I/O)-realizability, in the sense that if q_0 is a winning state for the system in $G_{\text{SYN}}(\mathcal{A}, I, O)$, then $\mathcal{A}$ is (I/O)-realizable. However, there are (I/O)-realizable NBWs for which q_0 is not winning for Player OR. The inherent difficulty in $\mathcal{A}$ being nondeterministic lies in the fact that each move of Player OR to a successor state in $\mathcal{A}$ should accommodate all possible future choices of Player AND. If different future choices of Player AND induce computations that are all in the language of $\mathcal{A}$ yet require different nondeterministic choices, then Player OR cannot win.

3 (I/O)-Aware Good-for-Games Automata

For an automaton $\mathcal{A}$ over $2^{I \cup O}$ and a word $w_I \in (2^I)^\omega$, we say that w_I is *hopeful in* $\mathcal{A}$ if there exists a word $w_O \in (2^O)^\omega$ such that $w_I \oplus w_O \in L(\mathcal{A})$. Consider a nondeterministic automaton $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$, and let $g : (2^I)^* \to 2^O \times Q$ be a function. We denote the first and second components of g by g_O and g_Q, respectively. That is, for a word $w_I \in (2^I)^*$, we have that $g(w_I) = \langle g_O(w_I), g_Q(w_I) \rangle$. We say that $\mathcal{A}$ is (I/O)-*aware GFG* if there exists a function $g : (2^I)^* \to 2^O \times Q$ such that the following hold.

1. $g(\epsilon) = \langle \emptyset, q_0 \rangle$,
2. The strategy g is compatible with δ. Thus for every $w_I \in (2^I)^*$ and $i \in 2^I$, we have that $g_Q(w_I \cdot i) \in \delta(g_Q(w_I), i \cup g_O(w_I))$, and
3. The strategy g "covers" all input words that are hopeful in $\mathcal{A}$. Thus for every w_I that is hopeful in $\mathcal{A}$, we have that $g_Q(w_I) = g_Q(\epsilon), g_Q(i_0), g_Q(i_0 \cdot i_1), \ldots$ is an accepting run on $w_I \oplus (g_O(\epsilon) \cdot g_O(i_0) \cdot g_O(i_0 \cdot i_1) \cdots)$.

Example 1. Consider the nondeterministic Büchi (in fact, looping) automaton $\mathcal{A}_2$ over $2^{\{a,b,c\}}$ appearing in Figure 2. Missing transitions lead to a rejecting sink.

Note that $\mathcal{A}_2$ is not GFG. Indeed, a strategy $g : (2^{\{a,b,c\}})^* \to Q$ neglects either the word $\{a, b\}^\omega$, in the case $g(\{a, b\}) = q_1$, or the word $\{a, b\} \cdot \{a, b, c\}^\omega$, in the case $g(\{a, b\}) = q_2$.

Assume that a and b are input signals and c is an output signal. Now, a function $g : (2^{\{a,b\}})^* \to 2^{\{c\}} \times Q$ such that $g(\epsilon) = \langle \emptyset, q_0 \rangle$, $g(\{a, b\}) = \langle \emptyset, q_2 \rangle$, $g(\{a\}) = \langle \emptyset, q_3 \rangle$

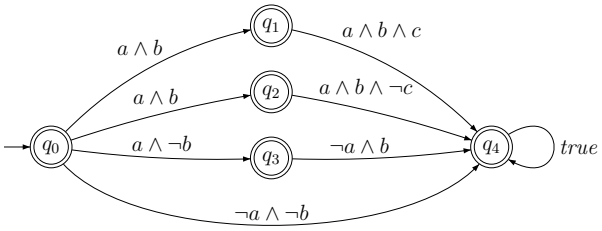

Fig. 2. The automaton $\mathcal{A}_2$ is not GFG, yet is $(\{a,b\}/\{c\})$-aware GFG.

and $g(\emptyset \cdot (2^{\{a,b\}})^*) = g(\{a\} \cdot \{b\} \cdot (2^{\{a,b\}})^*) = g(\{a,b\} \cdot \{a,b\} \cdot (2^{\{a,b\}})^*) = \langle \emptyset, q_4 \rangle$, witnesses that $\mathcal{A}_2$ is $(\{a,b\}/\{c\})$-aware GFG. Intuitively, identifying c as an output enables $\mathcal{A}_2$ to accept only one of the words $\{a,b\}^\omega$ and $\{a,b\} \cdot \{a,b,c\}^\omega$. □

As demonstrated in Example 1, the distinction between the nondeterminism in I and O makes some automata good for games. Beyond being applicable to more specifications, the cooperative nature of the nondeterminism in O makes (I/O)-aware GFG automata unboundedly more succinct than GFG automata. Indeed, a GFG automaton may need a large state space in order to recognize all computations, whereas an (I/O)-aware GFG automaton only needs to cover all hopeful input words. Similarly, determinization, even in GFG automata, may be needed in order to accept in an on-line manner all computations, and is not needed if we only care to accept a subset of them. We get back to this point in Section 7.

Remark 1. **[On (I/O)-aware DBP automata]** Translating LTL formulas to nondeterministic automata, one typically uses the Büchi acceptance condition [23], which has motivated our focus on NBWs. For safety properties, one ends up with NLWs – nondeterministic looping automata. As studied in [16, 4], GFG NLWs are *determinizable by prunning* (DBP, for short); that is, their transition function embodies a deterministic transition function that recognizes the same language. Extending the study to the (I/O)-aware setting is possible: We say that an NBW $\mathcal{A}$ is (I/O)*-aware DBP* if it is possible to resolve the nondeterministic choices of $\mathcal{A}$ by choosing, for every state q and assignment $i \in 2^I$, a transition from q that agrees with i, in a way that covers all input sequences that are hopeful in $\mathcal{A}$. In Appendix A we define (I/O)-aware DBP automata formally and show that all the known results about DBPness extend easily to the (I/O)-aware setting. In particular, while (I/O)-aware DBP automaton is (I/O)-aware GFG, the reverse direction is valid only for (I/O)-aware GFG NLWs. □

4 Synthesis with (I/O)-Aware GFG Automata

In this section we show that, as has been the case with GFG automata, (I/O)-aware GFG automata are sound and complete for (I/O)-realizability. Also, in spite of their succinctness, the complexity of the synthesis problem for (I/O)-aware GFG automata coincides with that of deterministic automata, and the problem of deciding whether a given automaton is (I/O)-aware GFG is not more complex than the problem of deciding whether a given automaton is GFG.

Theorem 1. (I/O)*-aware GFG automata are sound and complete for* (I/O)*-realizability.*

Proof. Consider an (I/O)-aware GFG automaton $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$. We show that q_0 is winning for Player OR (the system) in $G_{\text{SYN}}(\mathcal{A}, I, O)$ iff $\mathcal{A}$ is (I/O)-realizable.

The first direction holds already for general automata: if Player OR wins $G_{\text{SYN}}(\mathcal{A}, I, O)$, then his winning strategy induces an (I/O)-transducer that (I/O)-realizes $\mathcal{A}$.

For the other direction, assume that $\mathcal{A}$ is a realizable (I/O)-aware GFG automaton. Let $g : (2^I)^* \to 2^O \times Q$ be a function that witnesses $\mathcal{A}$'s (I/O)-aware GFGness, and let $G_{\text{SYN}}(\mathcal{A}, I, O) = \langle Q, Q \times 2^I, E, \alpha \rangle$ be the synthesis game. We describe a winning strategy for Player OR in $G_{\text{SYN}}(\mathcal{A}, I, O)$. Consider a prefix of a play $q_0, \langle q_0, i_0 \rangle, q_1, \langle q_1, i_1 \rangle, \ldots, q_k, \langle q_k, i_k \rangle$. A winning strategy for Player OR is to move from $\langle q_k, i_k \rangle$ to $g_Q(i_0 \cdot i_1 \cdots i_k)$. By the second condition on (I/O)-aware GFGness, the function g is compatible with δ, and so the above move exists. In addition, since $\mathcal{A}$ is realizable, then all the words in $(2^I)^\omega$ are hopeful in $\mathcal{A}$. Then, by the third condition on (I/O)-aware GFGness, we have that the run that g_Q produces is accepting. Therefore, this strategy is indeed winning for Player OR. □

Example 2. Consider again the automaton $\mathcal{A}_2$ from Example 1. Let $I = \{a, b\}$ and $O = \{c\}$. It is easy to see that Player OR does not win the synthesis game on $\mathcal{A}_2$. Indeed, Player AND can win by starting with input $\{b\}$. Since $\mathcal{A}_2$ is $(\{a, b\}/\{c\})$-aware GFG, we can conclude that $\mathcal{A}_2$ is not $(\{a, b\}/\{c\})$-realizable. □

The following corollary follows immediately from Theorem 1 and from the known complexity of deciding Büchi games [22].

Corollary 1. *The synthesis problem for (I/O)-aware automata with acceptance condition γ can be solved in the complexity of deciding games with winning condition γ. In particular, the synthesis problem for (I/O)-aware NBWs can be solved in quadratic time.*

We turn to study the complexity of deciding whether a given automaton is (I/O)-aware GFG. For an NBW $\mathcal{A} = \langle 2^X, Q, q_0, \delta, \alpha \rangle$ and a partition $\langle I, O \rangle$ of X, we define the *projection* of $\mathcal{A}$ on I as $\mathcal{A}_{|I} = \langle 2^I, Q, q_0, \delta', \alpha \rangle$, where for every two states $q_1, q_2 \in Q$ and letter $i \in 2^I$, we have that $q_2 \in \delta'(q_1, i)$ iff there exists $o \in 2^O$ such that $q_2 \in \delta(q_1, i \cup o)$. Thus, $\mathcal{A}_{|I}$ is obtained from $\mathcal{A}$ by hiding the 2^O-component in its transitions. Note that $\mathcal{A}_{|I}$ accepts all the words in $(2^I)^\omega$ that are hopeful in $\mathcal{A}$.

Lemma 1. *For every automaton $\mathcal{A}$ over 2^X and every partition $\langle I, O \rangle$ of X, we have that $\mathcal{A}$ is (I/O)-aware GFG iff $\mathcal{A}_{|I}$ is GFG.*

Lemma 1 implies the following. The result for NBWs also uses [3].

Corollary 2. *Consider an acceptance condition γ. The complexity of deciding (I/O)-aware GFGness for γ automata coincides with the complexity of deciding GFGness for γ automata. In particular, deciding (I/O)-aware GFGness for NBWs can be done in polynomial time.*

5 Non-Monotonicity of (I/O)-Aware GFGness

For a set X of signals and two partitions $\langle I, O \rangle$ and $\langle I', O' \rangle$ of X to input and output signals, we say that $\langle I, O \rangle$ *has less control* than $\langle I', O' \rangle$ if $I' \subseteq I$ (equivalently, $O' \supseteq O$). That is, in $\langle I', O' \rangle$, the system assigns values to all the signals in O and possibly some more, which have been assigned by the environment in the partition $\langle I, O \rangle$.

Consider a specification NBW $\mathcal{A}$ over the alphabet 2^X. It is not hard to see that if $\langle I, O\rangle$ has less control than $\langle I', O'\rangle$ and $\mathcal{A}$ is (I/O)-realizable, then $\mathcal{A}$ is also (I'/O')-realizable. Indeed, an (I'/O')-transducer that realizes $\mathcal{A}$ can be obtained from an (I/O)-transducer that realizes $\mathcal{A}$ by moving the signals in $I \setminus I'$ from the transitions to the output function, and arbitrarily pruning nondeterminism. On the other hand, as we shall see below, the NBW $\mathcal{A}$ may be (I/O)-aware GFG and not (I'/O')-aware GFG, and vice-versa. We first study the two extreme partitions, namely the ones when the system has all or no control.

Lemma 2. *Every NBW over 2^X is $(\emptyset/X)$-aware GFG. Every NBW over 2^X is $(X/\emptyset)$-aware GFG iff it is GFG.*

We continue to the general case, showing that the identification of signals as input or output can both improve and harm (I/O)-aware GFGness. We start with an example.

Example 3. Consider again the NLW $\mathcal{A}_2$ from Example 1. As we have seen there, $\mathcal{A}_2$ is not GFG, yet is $(\{a, b\}/\{c\})$-aware GFG. Here, we claim that $\mathcal{A}_2$ is not $(\{a\}/\{b, c\})$-aware GFG. Thus, identifying b as an output signal harms (I/O)-aware GFGness. Indeed, a function that attempts to witness $(\{a\}/\{b, c\})$-aware GFGness is $g : (2^{\{a\}})^* \to 2^{\{b,c\}} \times Q$, and it neglects either the word $\{a\}^\omega$, in the case $g_Q(\{a\}) = q_3$, or the word $\{a\} \cdot \emptyset^\omega$, in the case $g_Q(\{a\}) \in \{q_1, q_2\}$. □

In Theorem 2 below we generalize Example 3 and show that the alternation between positive and negative effect of control on (I/O)-aware GFGness is unboundedly long. The proof can be found in Appendix C.3.

Theorem 2. [(I/O)-aware GFGness is not monotone] *For every $k \geq 1$, we can define a DLW $\mathcal{A}^k$ over an alphabet $2^{\{x_1,x_2\ldots,x_{2k}\}}$, such that for all $1 \leq j \leq k$, we have that $\mathcal{A}^k$ is not $(\{x_1, \ldots, x_{2j-1}\}/\{x_{2j}, \ldots, x_{2k}\})$-aware GFG and is $(\{x_1, \ldots, x_{2j}\}/\{x_{2j+1}, \ldots, x_{2k}\})$-aware GFG.*

When, however, the specification is (I/O)-realizable, then monotonicity holds for partitions that have less control than $\langle I, O\rangle$:

Theorem 3. *Consider an (I/O)-realizable NBW $\mathcal{A}$ and a partition $\langle I', O'\rangle$ that has less control than $\langle I, O\rangle$. If $\mathcal{A}$ is (I'/O')-aware GFG, then $\mathcal{A}$ is also (I/O)-aware GFG.*

Proof. Consider an (I/O)-realizable NBW $\mathcal{A} = \langle 2^{I\cup O}, Q, q_0, \delta, \alpha\rangle$, and let $\langle I', O'\rangle$ be a partition that has less control than $\langle I, O\rangle$. That is, $I \subseteq I'$. Assume that $\mathcal{A}$ is (I'/O')-aware GFG and let $g' : (2^{I'})^* \to 2^{O'} \times Q$ and $f : (2^I)^* \to 2^O$ be functions that witness $\mathcal{A}$'s (I'/O')-aware GFGness and (I/O)-realizability, respectively. We define a function $g : (2^I)^* \to 2^O \times Q$ that witnesses $\mathcal{A}$'s (I/O)-aware GFGness.

Essentially, we construct g as follows. Given a word $w_I \in (2^I)^*$, we first use f in order to extend w_I to a word over $2^{I'}$, and then apply g' on the extended word. Formally, for $w_I = i_0, i_1, \cdots \in (2^I)^*$, let $w_O = f(i_0), f(i_0 \cdot i_1), \cdots \in (2^O)^*$. That is, w_O is the word that f pairs with w_I. Let $C = O \setminus O'$ be the set of signals for which control is lost in the transition from the partition $\langle I, O\rangle$ to $\langle I', O'\rangle$. Let $u \in (2^C)^*$ be the projection of w_O on C. That is, $u = f(i_0) \cap C, f(i_0 \cdot i_1) \cap C, \cdots \in (2^C)^*$. Note that $C = I' \setminus I$. Thus, we can define $w_{I'} = w_I \oplus u \in (2^{I'})^*$, and we define $g(w_I) = \langle g'_O(w_{I'}) \cup c, g'_Q(w_{I'})\rangle$, where c is the last letter of u.

Since $\mathcal{A}$ is (I'/O')-aware GFG, the function g' induces an accepting run of $\mathcal{A}$ on every word in $(2^{I'})^\omega$ that is hopeful in $\mathcal{A}$. This holds also for the word generated above, and so the function g witnesses $\mathcal{A}$'s (I/O)-aware GFGness. □

6 (I^+/O^-)-Aware Good-for-Games Automata

The non-monotonicity of the behavior of the signals discussed in Section 5 points to a bothering situation. In particular, an automaton may be GFG, in fact even deterministic, and hence be sound and complete for (I/O)-realizability, and still not be (I/O)-aware GFG. In this section we address this by defining (I^+/O^-)-*aware GFG automata*, which consider, given $\mathcal{A}$, all the partitions of $I \cup O$ with respect to which $\mathcal{A}$ is sound and complete for (I/O)-realizability.

For an NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$, we say that $\mathcal{A}$ is (I^+/O^-)-*aware GFG* if there exists a partition $\langle I', O' \rangle$ that has less control than $\langle I, O \rangle$, namely $I \subseteq I'$, such that $\mathcal{A}$ is (I'/O')-aware GFG. Note that every (I/O)-aware GFG automaton is (I^+/O^-)-aware GFG. In fact, by Lemma 2, every GFG automaton is (I^+/O^-)-aware GFG. However, as Lemma 2 implies, there are (I^+/O^-)-aware GFG automata that are not GFG or not (I/O)-aware GFG. We first argue that (I^+/O^-)-aware GFG automata are sound and complete for (I/O)-realizability.

Theorem 4. *(I^+/O^-)-aware GFG automata are sound and complete for (I/O)-realizability.*

Proof. Let $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$ be an (I^+/O^-)-aware GFG automaton, and let $\langle I', O' \rangle$ be a partition that has less control than $\langle I, O \rangle$ and for which $\mathcal{A}$ is (I'/O')-aware GFG. We show that Player OR wins $G_{\text{SYN}}(\mathcal{A}, I, O)$ iff $\mathcal{A}$ is (I/O)-realizable. The first direction holds for general NBWs. For the other direction, assume that $\mathcal{A}$ is (I/O)-realizable. Then, as $\langle I', O' \rangle$ has less control than $\langle I, O \rangle$, Theorem 3 implies that $\mathcal{A}$ is (I/O)-aware GFG. Therefore, by Theorem 1, Player OR wins $G_{\text{SYN}}(\mathcal{A}, I, O)$. □

Theorem 4 enables us to extend Corollary 1 to (I^+/O^-)-aware GFG automata:

Corollary 3. *The synthesis problem for (I^+/O^-)-aware automata with acceptance condition γ can be solved in the complexity of deciding games with winning condition γ. In particular, the synthesis problem for (I^+/O^-)-aware NBWs can be solved in quadratic time.*

We turn to study the complexity of deciding (I^+/O^-)-aware GFGness. As we shall see, the non-monotonicity suggests that one should check all possible partitions of $I \cup O$. Formally, we have the following.

Theorem 5. *Consider an NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$. The problem of deciding whether $\mathcal{A}$ is (I^+/O^-)-aware GFG is polynomial in $|Q|$ and NP-complete in $|I \cup O|$.*

Proof. Given an NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$, one can decide whether $\mathcal{A}$ is (I^+/O^-)-aware GFG by checking, for all partitions $\langle I', O' \rangle$ that have less control than $\langle I, O \rangle$, whether $\mathcal{A}$ is (I'/O')-aware GFG. By Corollary 2, each such check can be done in time polynomial in $|Q|$. Further, by guessing a partition $\langle I', O' \rangle$ that has less control than $\langle I, O \rangle$, and checking whether $\mathcal{A}$ is (I'/O')-aware GFG we get membership in NP with respect to $|I \cup O|$.

We proceed to the lower bound, showing that deciding (I^+/O^-)-aware GFGness is NP-hard in $|I \cup O|$. The 3SAT problem, known to be NP-hard, is to decide whether a given 3CNF formula is satisfiable. We show a reduction from 3SAT to deciding (I^+/O^-)-aware GFGness. Let $X = \{x_1, \ldots, x_n\}$ be a set of variables, and consider a 3CNF formula $\varphi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, where for every $1 \leq j \leq m$, we have that $c_j = l_1^j \vee l_2^j \vee l_3^j$, and $l_k^j \in \bigcup_{x \in X} \{x, \neg x\}$, for all $1 \leq k \leq 3$. We construct an NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, Q \rangle$

such that $\mathcal{A}$ is (I^+/O^-)-aware GFG iff φ is satisfiable. In fact, all the states of $\mathcal{A}$ are accepting, thus the lower bound holds already for NLWs.

We define $I = \{1, \ldots, n+m\}$ and $O = \bigcup_{x \in X} \{x, \tilde{x}\}$. For every clause c, we define $\tilde{c}$ as the clause obtained by replacing every literal of the form $\neg x$ with the signal $\tilde{x}$. For example, if $c = x_1 \vee \neg x_2 \vee x_3$, then $\tilde{c} = x_1 \vee \tilde{x}_2 \vee x_3$. We define, for every $1 \leq j \leq n$, the NLW $\mathcal{B}^j = \langle 2^{I \cup O}, P_j, p_0^j, \delta_j^{\mathcal{B}}, P_j \rangle$, illustrated in Figure 3. In addition, for every $1 \leq j \leq m$, we define the NLW $\mathcal{C}^j = \langle 2^{I \cup O}, S_j, s_0^j, \delta_j^{\mathcal{C}}, S_j \rangle$, illustrated in Figure 3.

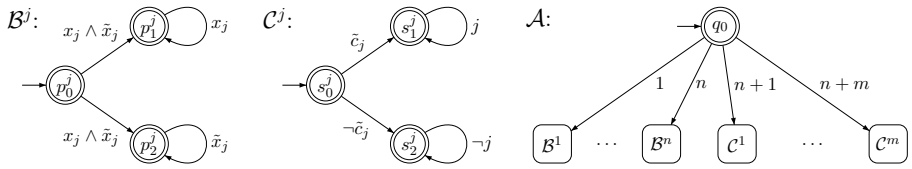

Fig. 3. The NLWs $\mathcal{B}^j$, $\mathcal{C}^j$, and $\mathcal{A}$.

The NBW $\mathcal{A}$ combines $\mathcal{B}^1, \ldots, \mathcal{B}^n, \mathcal{C}^1, \ldots, \mathcal{C}^m$ in the way illustrated in Figure 3. That is, $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, Q \rangle$, where $Q = \{q_0\} \cup (\bigcup_{1 \leq j \leq n} P_j) \cup (\bigcup_{1 \leq j \leq m} S_j)$, and for all $q \in Q$ and $\sigma \in 2^{I \cup O}$, we have that

$$\delta(q, \sigma) = \begin{cases} \{p_0^j\} & \text{if } q = q_0 \text{ and } \sigma = \{j\}, \text{ for } 1 \leq j \leq n \\ \{s_0^{j-n}\} & \text{if } q = q_0 \text{ and } \sigma = \{j\}, \text{ for } n+1 \leq j \leq n+m \\ \delta_j^{\mathcal{B}}(q, \sigma) & \text{if } q \in P_j, \text{ for } 1 \leq j \leq n \\ \delta_j^{\mathcal{C}}(q, \sigma) & \text{if } q \in S_j, \text{ for } 1 \leq j \leq m \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that both $|I \cup O|$ and $|Q|$ are of size polynomial in $|\varphi|$. In Appendix C.4, we prove that $\mathcal{A}$ is (I^+/O^-)-aware GFG iff φ is satisfiable. □

7 (I/O)-Awareness and Synthesis

A natural problem that follows from our results is the generation of small (I^+/O^-)-aware GFG automata. Thus, given an NBW $\mathcal{A}$, return a minimal (I^+/O^-)-aware GFG automaton equivalent to $\mathcal{A}$. In this section we argue that the equivalence requirement is too strong and can be relaxed to produce even smaller automata. We relate the problem of generating (I/O)-aware GFG automata with that of generating realizing (I/O)-transducers, and show how it sheds light on an important open problem, namely whether minimal realizing transducers for NBW specifications with n states need $2^{O(n \log n)}$ or only $2^{O(n)}$ states. We also suggest a heuristic algorithm that solves synthesis given a specification and its negation, possibly avoiding determinization and getting a transducer with less than $2^{O(n \log n)}$ states.

7.1 Minimal (I/O)-Transducers

We start with the definition of *covering* automata, which replaces the equivalence condition. For two automata $\mathcal{A}$ and $\mathcal{A}'$ over $2^{I \cup O}$, we say that $\mathcal{A}'$ (I/O)-*covers* $\mathcal{A}$ if $L(\mathcal{A}') \subseteq L(\mathcal{A})$

and $L(\mathcal{A}_{|I}) = L(\mathcal{A}'_{|I})$. Thus, every word in $(2^I)^\omega$ that is hopeful in $\mathcal{A}$ is hopeful also in $\mathcal{A}'$, and $\mathcal{A}'$ does not extend the language of $\mathcal{A}$. [3] Moreover, $\mathcal{A}'$ *(I^+/O^-)-covers* $\mathcal{A}$ if there is a partition $\langle I', O'\rangle$ that has less control than $\langle I, O\rangle$ such that $L(\mathcal{A}') \subseteq L(\mathcal{A})$ and $L(\mathcal{A}_{|I'}) = L(\mathcal{A}'_{|I'})$. We then say that $\mathcal{A}'$ (I^+/O^-)-covers $\mathcal{A}$ with $\langle I', O'\rangle$. Note that if $\mathcal{A}'$ (I/O)-covers $\mathcal{A}$, then $\mathcal{A}'$ also (I^+/O^-)-covers $\mathcal{A}$ (with the partition $\langle I, O\rangle$), yet, as has been the case with (I^+/O^-)-awareness, allowing coverage with partitions with less control strictly strengthen the definition, and, as we show below, is still sound and complete for (I/O)-realizability.

Theorem 6. *Consider two automata $\mathcal{A}$ and $\mathcal{A}'$ over $2^{I\cup O}$. If $\mathcal{A}'$ (I^+/O^-)-covers $\mathcal{A}$ with $\langle I', O'\rangle$ and is (I'/O')-aware GFG, then $\mathcal{A}'$ is sound and compete for (I/O)-realizability of $\mathcal{A}$.*

Proof. We prove that Player OR wins $G_{\text{SYN}}(\mathcal{A}', I, O)$ iff $\mathcal{A}$ is (I/O)-realizable. The first direction is easy: if Player OR wins $G_{\text{SYN}}(\mathcal{A}', I, O)$, then $\mathcal{A}'$ is (I/O)-realizable. Since $L(\mathcal{A}') \subseteq L(\mathcal{A})$, then every transducer that (I/O)-realizes $\mathcal{A}'$ also (I/O)-realizes $\mathcal{A}$, thus $\mathcal{A}$ is (I/O)-realizable.

For the other direction, assume that $\mathcal{A}$ is (I/O)-realizable, and let $\mathcal{T}$ be a transducer that (I/O)-realizes $\mathcal{A}$. Consider a word $w_I \in (2^I)^\omega$. Let $w_{I'} = \mathcal{T}(w_I) \cap (2^{I'})^\omega$, that is, $w_{I'}$ is the projection on I' of the computation of $\mathcal{T}$ on w_I. Clearly, $w_{I'} \in L(\mathcal{A}_{|I'})$, and therefore, $w_{I'} \in L(\mathcal{A}'_{|I'})$. Let $g : (2^{I'})^* \to 2^{O'} \times Q$ be a function that witnesses that $\mathcal{A}'$ is (I'/O')-aware GFG. We describe a winning strategy for Player OR in $G_{\text{SYN}}(\mathcal{A}', I, O)$. Recall that in $G_{\text{SYN}}(\mathcal{A}', I, O)$, Player OR responds to a sequence of input letters over 2^I with an output letter in 2^O and an according transition. Essentially, Player OR uses $\mathcal{T}$ in order to extend a given sequence of input letters in $(2^I)^*$ to a sequence of letters in $(2^{I'})^*$, and then plays accordingly to g. Formally, we extend the notion of computations of $\mathcal{T}$ to finite words. Consider a prefix of a play $q_0, \langle q_0, i_0\rangle, q_1, \langle q_1, i_1\rangle, \ldots, q_k, \langle q_k, i_k\rangle$. A winning strategy for Player OR in $G_{\text{SYN}}(\mathcal{A}', I, O)$ is to move from $\langle q_k, i_k\rangle$ to $g_Q((2^{I'})^* \cap \mathcal{T}(w_I))$, where $w_I = i_0 \cdot i_1 \cdots i_k$. Recall that for all $w_I \in (2^I)^\omega$, we have that $w_{I'}$ is hopeful in $\mathcal{A}'$. Therefore, the above strategy is winning for Player OR. □

Theorem 6 implies that one can solve synthesis for an automaton $\mathcal{A}$ by constructing an (I'/O')-aware GFG automaton $\mathcal{A}'$ that (I^+/O^-)-covers $\mathcal{A}$ with $\langle I', O'\rangle$, rather than an equivalent one, and solving $G_{\text{SYN}}(\mathcal{A}', I, O)$.

Remark 2. Note that for $\mathcal{A}'$ to (I^+/O^-)-cover $\mathcal{A}$ with $\langle I', O'\rangle$, the $L(\mathcal{A}_{|I}) = L(\mathcal{A}'_{|I})$ requirement is strengthened to $L(\mathcal{A}_{|I'}) = L(\mathcal{A}'_{|I'})$. This is crucial. That is, it may be the case that $\mathcal{A}$ is realizable, yet Player OR losses $G_{\text{SYN}}(\mathcal{A}', I, O)$ for an (I^+/O^-)-aware GFG automaton $\mathcal{A}'$ such that $L(\mathcal{A}_{|I}) = L(\mathcal{A}'_{|I})$ and $L(\mathcal{A}') \subseteq L(\mathcal{A})$. As an example, consider an automaton $\mathcal{A}$ over $2^{\{a,b,c\}}$ with $L(\mathcal{A}) = (2^{\{a,b,c\}})^\omega$, and the automaton $\mathcal{A}_2$ from Example 1. Let $I = \{a\}$ and $O = \{b, c\}$. It is easy to see that all the words in $(2^{\{a\}})^\omega$ are hopeful in $\mathcal{A}_2$, and that $L(\mathcal{A}_2) \subseteq L(\mathcal{A})$. In addition, $\mathcal{A}$ is clearly realizable. Recall that $\mathcal{A}_2$ is $(\{a, b\}/\{c\})$-aware GFG, thus it is (I^+/O^-)-aware GFG. Yet, Player OR looses $G_{\text{SYN}}(\mathcal{A}_2, I, O)$. Indeed, a winning strategy for Player AND is to start with input $\{a\}$, and then choose input $\emptyset$ if Player OR responds with output in which b is true, and choose input $\{a\}$ otherwise. □

[3] Note that the definition is different than *open implication* in [10], where $\mathcal{A}'$ *open implies* $\mathcal{A}$ if every (I/O)-transducer that (I/O)-realizes $\mathcal{A}'$ also (I/O)-realizes $\mathcal{A}'$. For example, an empty $\mathcal{A}'$ open implies every unrealizable $\mathcal{A}$, yet need not (I/O)-cover it.

We turn to study the size of a minimal (I/O)-covering (I/O)-aware GFG NBW. Since unboundedly large parts of the specification automaton may not bee needed for its realization, we have the following.

Lemma 3. *Consider an NBW $\mathcal{A}$. An (I/O)-aware GFG NBW that (I/O)-covers $\mathcal{A}$ may be unboundedly smaller than any GFG automaton equivalent to $\mathcal{A}$.*

Theorem 7. *Consider an (I/O)-realizable NBW $\mathcal{A}$. The size of a minimal (I/O)-aware GFG NBW that (I/O)-covers $\mathcal{A}$ coincides with the size of a minimal (I/O)-transducer that (I/O)-realizes $\mathcal{A}$.*

Proof. It is easy to see that if $\mathcal{A}$ is realizable, then an (I/O)-transducer that (I/O)-realizes $\mathcal{A}$ can be viewed as an (I/O)-aware GFG NLW that (I/O)-covers $\mathcal{A}$. Conversely, a winning strategy for the system on $G_{\text{SYN}}(\mathcal{A}, I, O)$, for an (I/O)-aware GFG NBW $\mathcal{A}$, can be viewed as an (I/O)-transducer that (I/O)-realizes $\mathcal{A}$. Note that since our definition of (I/O)-transducers has the output assignments in the states, we actually need $2^{|O|}$ copies of each state. These copies, however, are not needed if one considers (I/O)-transducers with output assignments on the transitions. □

We continue to the problem of generating small covering (I/O)-aware GFG NBWs. By Theorem 7, the latter coincides with the problem of generating small realizing transducers. The currently known upper bound for the size of a realizing transducer, starting with a specification NBW $\mathcal{A}$ with n states, is $2^{O(n \log n)}$, and is based on playing the synthesis game on a deterministic automaton equivalent to $\mathcal{A}$. Unlike the case of determinization, no matching lower bound is known. Below we relate the existence of such a lower bound with the existence of an NBW that is easy to complement yet hard to determinize.

Theorem 8. *Let $n \geq 1$. If there is an NBW $\mathcal{A}_n$ with n states such that (1) $\mathcal{A}_n$ is easy to complement: there is an NBW $\mathcal{A}'_n$ with $O(n)$ states such that $L(\mathcal{A}'_n) = \Sigma^\omega \setminus L(\mathcal{A}_n)$, yet (2) $\mathcal{A}_n$ is hard to determinize: a DBW equivalent to $\mathcal{A}_n$ needs at least $2^{O(n \log n)}$ states, then there is a realizable NBW $\mathcal{B}_n$ with $O(n)$ states such that the minimal realizing transducer for $\mathcal{B}_n$ needs at least $2^{O(n \log n)}$ states.*

Proof. Let Σ be the alphabet of $\mathcal{A}_n$. We define $\mathcal{B}_n$ over $\Sigma \times \{0,1\}$ so that $L(\mathcal{B}_n)$ contains all words $w \oplus v$ such that $w \in L(\mathcal{A}_n)$ iff v has infinitely many 1's. Thus, the projection on Σ is in $L(\mathcal{A}_n)$ iff the projection on $\{0,1\}$ has infinitely many 1's.

It is not hard to see that we can define $\mathcal{B}_n$ by an NBW with $O(n)$ states. Indeed, we can define $\mathcal{B}_n$ as the union of an NBW $\mathcal{B}^1_n$ for words $w \oplus v$ such that $w \in L(\mathcal{A}_n)$ and v has infinitely many 1's and an NBW $\mathcal{B}^2_n$ for words $w \oplus v$ such that $w \notin L(\mathcal{A}_n)$ and v has finitely many 1's. The NBW $\mathcal{B}^1_n$ is the product of $\mathcal{A}_n$, which has n states, with a 2-state DBW for "infinitely many 1's", so its size is $O(n)$. The NBW $\mathcal{B}^2_n$ is the product of an NBW that complements $\mathcal{A}_n$, and which, by Condition (1), has $O(n)$ states, with a 3-state NBW for "only finitely many 1's". So the size of $\mathcal{B}^2_n$ is also $O(n)$.

Now, if we view Σ as an input alphabet and view $\{0,1\}$ as an output alphabet, then a $(\Sigma/\{0,1\})$-transducer for $\mathcal{B}_n$ induces a DBW for $\mathcal{A}$ of the same size (note we refer here to Σ and $\{0,1\}$ as input and output alphabets, rather than signals, but this is a technical issue, as we could have encoded them). To see this, consider a $(\Sigma/\{0,1\})$-transducer $\mathcal{T}_n = \langle \Sigma, \{0,1\}, S, s_0, \rho, \tau \rangle$ that realizes $\mathcal{B}_n$, and let $\mathcal{D}_n = \langle \Sigma, S, s_0, \rho, \alpha \rangle$ be a DBW with $\alpha = \{s : \tau(s) = 1\}$. We claim that $L(\mathcal{D}_n) = L(\mathcal{A}_n)$. Indeed, since $\mathcal{T}_n$ realizes $\mathcal{B}_n$, then for every input word $w \in \Sigma^\omega$, the computation of $\mathcal{T}_n$ on w has infinitely many 1's iff $w \in L(\mathcal{A}_n)$.

Hence, the run of $\mathcal{D}_n$ on w visits α infinitely often iff $w \in L(\mathcal{A}_n)$. Hence, by Condition (2), the transducer $\mathcal{T}_n$ needs at least $2^{O(n \log n)}$ states. □

Remark 3. Theorem 8 refers to languages that are DBW-recognizable. It is easy to extend it to all ω-regular languages by considering automata with a parity acceptance condition of index k, for every $k \geq 1$. Then, the automata $\mathcal{B}_n$ are over the alphabet $\Sigma \times \{1, \ldots, k\}$, and they accept all words $w \oplus v$ such that $w \in L(\mathcal{A}_n)$ iff the minimal index that appears in v infinitely often is even. □

7.2 Using Over-Approximating GFG automata

In Section 7.1, we suggest the use of GFG automata that under-approximate the specification. In this section we suggest a heuristic that starts with an LTL formula φ and is based on GFG automata that over-approximate NBWs for φ and $\neg\varphi$. The GFG automaton that over approximates an NBW $\mathcal{A}$ is obtained by applying the subset construction on $\mathcal{A}$ and adding nondeterministic transitions to states associated with strict subsets of the successor subset. By working with the over-approximations of both φ and $\neg\varphi$, we iteratively refine the subset construction, adding information that makes the state spaces closer to that of DPWs for φ and $\neg\varphi$. This continues until we get a transducer that realizes φ or $\neg\varphi$, typically much earlier than full determinization is performed.

We now describe the heuristic in more detail. For an NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$, the *nondeterministic subset construction* of $\mathcal{A}$ is $\text{NSC}(\mathcal{A}) = \langle 2^{I \cup O}, 2^Q, \{q_0\}, \delta', \alpha' \rangle$, where $S' \in \delta'(S, \sigma)$ iff $S' \subseteq \cup_{q \in S} \delta(q, \sigma)$, and $S \in \alpha'$ iff $S \cap \alpha \neq \emptyset$. That is, $\text{NSC}(\mathcal{A})$ extends the subset construction of $\mathcal{A}$ by adding transitions, for every $S \in 2^Q$ and $\sigma \in 2^{I \cup O}$, to all the subsets of $\delta(S, \sigma)$. Note that α' contains all sets whose intersection with α is not empty, and so $\text{NSC}(\mathcal{A})$ over-approximates $\mathcal{A}$, thus $L(\mathcal{A}) \subseteq L(\text{NSC}(\mathcal{A}))$. In addition, $\text{NSC}(\mathcal{A})$ is DBP, and so it is GFG with respect to the deterministic subset construction of $\mathcal{A}$.

Given an LTL formula φ, let $\mathcal{A}_\varphi$ and $\mathcal{A}_{\neg\varphi}$ be NBWs for φ and $\neg\varphi$, respectively. By determinacy of games, $\mathcal{A}_\varphi$ is (I/O)-realizable iff $\mathcal{A}_{\neg\varphi}$ is not (O/I)-realizable.[4] We use $\tilde{\mathcal{A}}_\varphi$ and $\tilde{\mathcal{A}}_{\neg\varphi}$ to denote the over-approximations of $\mathcal{A}_\varphi$ and $\mathcal{A}_{\neg\varphi}$, respectively, generated during the algorithm. Initially, $\tilde{\mathcal{A}}_\varphi = \text{NSC}(\mathcal{A}_\varphi)$ and $\tilde{\mathcal{A}}_{\neg\varphi} = \text{NSC}(\mathcal{A}_{\neg\varphi})$. In every iteration, we solve both $G_{\text{SYN}}(\tilde{\mathcal{A}}_\varphi, I, O)$ and $G_{\text{SYN}}(\tilde{\mathcal{A}}_{\neg\varphi}, O, I)$. Since we work with over-approximations, the following three outcomes are possible.

1. Player AND wins $G_{\text{SYN}}(\tilde{\mathcal{A}}_{\neg\varphi}, O, I)$. Then, we conclude that φ is realizable, the winning strategy for Player AND induces a transducer that realizes φ, and we are done.
2. Player AND wins $G_{\text{SYN}}(\tilde{\mathcal{A}}_\varphi, I, O)$. Then, we conclude that φ is not realizable, and we are done.
3. Player OR wins in both games. Note this is possible only due to the over-approximation. We model-check the single computation w that is the outcome of the interaction of the winning strategies of Player OR in the games. If $w \models \varphi$, we conclude that $\tilde{\mathcal{A}}_\varphi$ needs to be refined. At this point we may also model check the transducer induced by the winning strategy of Player OR in $G_{\text{SYN}}(\tilde{\mathcal{A}}_{\neg\varphi}, O, I)$, and conclude that φ is not realizable if the

[4] A more precise definition of the dual setting adds to the realizability notation the parameter of "who moves first". Then, $\mathcal{A}_\varphi$ is (I/O)-realizable with the environment moving first iff $\mathcal{A}_{\neg\varphi}$ is not (O/I)-realizable with the system (that is, the player that generates signals in O) moving first. Adding this parameter is easy, yet makes the writing more cumbersome, so we give it up.

transducer satisfies $\neg\varphi$. Dually, if $w \not\models \varphi$, we conclude that $\tilde{\mathcal{A}}_{\neg\varphi}$ needs to be refined, and we may model check the transducer induced by the winning strategy of Player OR in $G_{\text{SYN}}(\tilde{\mathcal{A}}_{\varphi}, I, O)$, and conclude it realizes φ. If model checking fails, or if we decide to skip it, we refine (possibly both $\mathcal{A}_{\varphi}$ and $\mathcal{A}_{\neg\varphi}$, one according to w and one with respect to the counterexample obtained from the model checking) and continue to the next iteration.

It is left to describe the refinement. Essentially, the refinement of $\tilde{\mathcal{A}}_{\varphi}$ (and similarly for $\tilde{\mathcal{A}}_{\neg\varphi}$) with respect to a counterexample word w excludes w from $\tilde{\mathcal{A}}_{\varphi}$, and is done in a way that eventually results in a GFG automaton for $\mathcal{A}_{\varphi}$, unless the procedure halts in an earlier iteration. The refinement may use any on-the-fly determinization construction whose state space consists of information on top of the subset construction (e.g., Safra trees [19] or reduced trees [8]). Let $\mathcal{D}_{\varphi}$ be a DPW for $\mathcal{A}_{\varphi}$, and let r be the run of $\mathcal{D}_{\varphi}$ on w. By the way we defined and have refined $\tilde{\mathcal{A}}_{\varphi}$ so far, the states in r can be mapped to the states of $\tilde{\mathcal{A}}_{\varphi}$. For example, in the first iteration, where the states of $\tilde{\mathcal{A}}_{\varphi}$ are subsets of states in $\mathcal{A}_{\varphi}$, we use the fact that each state in $\mathcal{D}_{\varphi}$ is associated with such a subset. We use this mapping in order to refine states of $\tilde{\mathcal{A}}_{\varphi}$ that are mapped to by different states along r, and update the acceptance condition of $\tilde{\mathcal{A}}_{\varphi}$ accordingly. See Appendix B for an example.

As with other counterexample-guided refinement methodologies, several heuristics concerning the choice of a counterexample are possible. Here, we also suggest heuristics for the choice of winning strategy in both $G_{\text{SYN}}(\tilde{\mathcal{A}}_{\varphi}, I, O)$ and $G_{\text{SYN}}(\tilde{\mathcal{A}}_{\neg\varphi}, O, I)$. This is where the GFGness of the nondeterministic subset construction plays a role. We say that a winning strategy for a player is *minimalistic* if for every state associated with a subset $S \subseteq 2^Q$ that she chooses, every state that is associated with a subset $S' \subset S$ is losing for her. By choosing minimalistic strategies, we avoid determinization associated with large sets, whenever possible. In fact, when $\mathcal{A}_{\varphi}$ is (I/O)-aware GFG, a winning strategy may coincide with the GFG strategy. In addition, in the case Player OR wins both of the games, we can consider several winning strategies, and either refine or check the induced transducer with respect to each one of them.

In the worst case, the algorithm halts when either $\tilde{\mathcal{A}}_{\varphi}$ or $\tilde{\mathcal{A}}_{\neg\varphi}$ is a DPW for φ or $\neg\varphi$, respectively. Thus, their size bounds the number of iterations. In each iteration, we solve two parity games, check whether a single computation satisfies φ, and optionally model-check a transducer – all these are done in less than exponential time, and so the overall time complexity is doubly exponential in $|\varphi|$, meeting the lower bound for LTL synthesis.

8 Discussion

We introduced (I^+/O^-)-aware GFG automata and studied their properties, especially in the context of synthesis. Our contribution significantly extends the type of automata that are sound and complete for (I/O)-realizability.

We left open the problem of generating minimal covering (I/O)-aware GFG automata, and show that it is tightly related, and in fact brings new insights, to the long-standing open problem of generating minimal (I/O)-realizing transducers, namely the problem of whether a minimal realizing transducer for an NBW specification with n states needs $2^{O(n \log n)}$ or only $2^{O(n)}$ states. The current upper bound goes through determinization and is $2^{O(n \log n)}$, but no matching lower bound is known. We related the problem to the one of finding NBWs that are easy to complement yet hard to determinize, which is also open. We believe that

this family of problems is of great practical interest, less because of the difference between $2^{O(n \log n)}$ and $2^{O(n)}$, and more because of the possibility, suggested by Theorem 8, that a simple determinization construction, or at least a simple synthesis algorithm, can be developed for the case NBWs are given for both the specification and its negation. Finally, we suggested a synthesis heuristic that uses easy to construct GFG automata for the specification and its negation: Determinization with a subset construction over-approximates the language, and GFGness under-approximates the language by allowing transitions to strict subsets. While the performance of the heuristic in practice still needs to be checked, we believe that the underlying idea, of combining a simple construction that extends the language with GFG nondeterministic choices that restrict the language, deserves further study.

References

1. B. Abu Radi and O. Kupferman. Minimizing GFG transition-based automata. In *Proc. 46th ICALP*, LIPIcs 132, 2019.
2. R. Alur, R. Bodik, G. Juniwal, M. Martin, M. Raghothaman, S. Seshia, R. Singh, A. Solar-Lezama E. Torlak, and A. Udupa. Syntax-Guided Synthesis. IEEE, 2013.
3. M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th FSTTCS*, LIPIcs 132, 2018.
4. U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. 40th ICALP*, LNCS 7966, 2013.
5. U. Boker, O. Kupferman, and M. Skrzypczak. How deterministic are Good-For-Games automata? In *Proc. 37th FSTTCS*, LIPIcs 93, 2017.
6. A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians*, 1963.
7. Th. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th ICALP*, LNCS 5556, 2009.
8. D. Fisman and Y. Lustig. A modular approach for Büchi determinization. In *Proc. 26th CONCUR*, 2015.
9. E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500, 2002.
10. K. Greimel, R. Bloem, B. Jobstmann, and M. Vardi. Open implication. In *Proc. 35th ICALP*, LNCS 5126, 2008.
11. T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th CSL*, LNCS 4207, 2006.
12. D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Proc. 42nd ICALP*, 2015.
13. O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.
14. O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th FOCS*, 2005.
15. K. Lehtinen and M. Zimmermann. Good-for-games ω-pushdown automata. In *Proc. 35th LICS*, 2020.
16. G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.
17. D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th STACS*, LNCS 1373, 1998.
18. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, 1989.
19. S. Safra. On the complexity of ω-automata. In *Proc. 29th FOCS*, 1988.
20. A. Solar-Lezama. Program Synthesis by Sketching. PhD thesis, UC Berkeley, 2008.
21. A. Solar-Lezama, L. Tancau, R. Bodik, S. A. Seshia, and V. Saraswat. Combinatorial sketching for finite programs. In *Proc. 12th ASPLOS*, 2006.
22. M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and Systems Science*, 32(2):182–221, 1986.
23. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *I&C*, 115(1):1–37, 1994.

A (I/O)-Aware Determinizability by Prunning

Translating LTL formulas to nondeterministic automata, one typically uses the Büchi acceptance condition [23], which has motivated our focus on NBWs. For safety properties, one ends up with nondeterministic looping automata, where all infinite runs are accepting. As studied in [16, 4], GFG NLWs are *determinizable by prunning* (DBP, for short); that is, their transition function embodies a deterministic transition function that recognizes the same language. In this section we study (I/O)-aware DBPness, showing that all known results about DBPness extend easily to the (I/O)-aware setting.

An NBW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$ is DBP if there is a DBW $\mathcal{A}' = \langle 2^{I \cup O}, Q, q_0, \delta', \alpha \rangle$ such that for all $q \in Q$ and $\sigma \in 2^{I \cup O}$, we have that $\delta'(q, \sigma) \subseteq \delta(q, \sigma)$ and $L(\mathcal{A}') = L(\mathcal{A})$. It is easy to see that every DBP NBW is GFG. Indeed, if $\mathcal{A}$ is DBP, then a function that witnesses its GFGness can follow the transitions of the deterministic automaton embodied in $\mathcal{A}$. The other direction does not hold: there are GFG NBWs that are not DBP [4]. For NLWs, however, GFGness does coincide with DBPness [16, 4].

Adding (I/O)-awareness to DBPness essentially means that it is possible to resolve the nondeterministic choices of $\mathcal{A}$ by choosing, for every state q and assignment $i \in 2^I$, a transition from q that agrees with i, in a way that covers all input sequences that are hopeful in $\mathcal{A}$. Formally, we say that $\mathcal{A}$ with a total transition function δ is (I/O)-*aware DBP* if there exists a function $g : Q \times 2^I \to 2^O \times Q$ such that the following hold.

- For every $q \in Q$ and $i \in 2^I$, we have that $g_Q(q, i) \in \delta(q, i \cup g_O(q, i))$, and
- For every word $w_I = i_0 \cdot i_1 \cdots \in (2^I)^\omega$, if w_I is hopeful in $\mathcal{A}$, then $q_0, q_1, q_2, \ldots$ is an accepting run on $w_I \oplus (o_0, o_1, o_2, \ldots)$, where for every index $j \geq 0$ we have that $q_{j+1} = g_Q(q_j, i_j)$ and $o_j = g_O(q_j, i_j)$.

We first argue that the connection between the (I/O)-aware GFGness of an automaton $\mathcal{A}$ and the GFGness of $\mathcal{A}_{|I}$ applies also for DBPness:

Lemma 4. *An automaton $\mathcal{A}$ is (I/O)-aware DBP iff $\mathcal{A}_{|I}$ is DBP.*

Proof. Assume first that $\mathcal{A}_{|I}$ is DBP and let $f : Q \times 2^I \to Q$ be the function that witnesses it. We extend f to a function $f' : Q \times 2^I \to 2^O \times Q$ as follows. For every $q \in Q$ and $i \in 2^I$, we define $f'(q, i) = \langle o, f(q, i) \rangle$, where $o \in 2^O$ is such that $f(q, i) \in \delta(q, i \cup o)$. Clearly, f' witnesses that $\mathcal{A}$ is (I/O)-aware DBP. For the second direction, assume that $\mathcal{A}$ is (I/O)-aware DBP and let $f : Q \times 2^I \to 2^O \times Q$ be the function that witnesses it. The function f_Q witnesses that $\mathcal{A}_{|I}$ is DBP. Indeed, f_Q produces accepting runs on all the words that are hopeful in $\mathcal{A}$, which are exactly the words in $L(\mathcal{A}_{|I})$. □

We continue and argue that the relation between DBPness and GFGness applies also in the (I/O)-aware setting:

Lemma 5. *1. Every (I/O)-aware DBP automaton is (I/O)-aware GFG.*
2. Every (I/O)-aware GFG NLW is (I/O)-aware DBP.
3. There exists an (I/O)-aware GFG NBW that is not (I/O)-aware DBP.

Proof. For the first claim, assume that $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$ is (I/O)-aware DBP, and let $f : Q \times 2^I \to 2^O \times Q$ be the function that witnesses this. We extend f to a function $g : (2^I)^* \to 2^O \times Q$, where $g(\epsilon) = \langle \emptyset, q_0 \rangle$, and for every $w_I \cdot i \in (2^I)^* \times 2^I$, we have that $g(w_I \cdot i) = f(g_Q(w_I), i)$. It is easy to see that g witnesses that $\mathcal{A}$ is (I/O)-aware GFG.

For the second claim, consider an (I/O)-aware GFG NLW $\mathcal{A}$. Then, by Lemma 1, we have that $\mathcal{A}_{|I}$ is GFG, and so, by [16, 4], we have that $\mathcal{A}_{|I}$ is DBP. Thus, by Lemma 4, the NLW $\mathcal{A}$ is (I/O)-aware DBP.

For the last claim, let $\mathcal{A}$ be a GFG NBW over some alphabet 2^X that is not DBP. By [4], such an NBW exists. By Lemma 2, we have that $\mathcal{A}$ is $(X/\emptyset)$-aware GFG. In addition, as $\mathcal{A} = \mathcal{A}_{|X}$, Lemma 4 implies that $\mathcal{A}$ is not $(X/\emptyset)$-aware DBP, as a function $f : Q \times 2^X \to 2^\emptyset \times Q$ that witnesses $\mathcal{A}$'s $(X/\emptyset)$-aware DBPness witnesses also its DBPness. □

Finally, the notion of (I^+/O^-)-aware GFG automata extends naturally to DBP, thus an NBW $\mathcal{A}$ is (I^+/O^-)-aware DBP iff there exists a partition $\langle I', O'\rangle$ that has less control than $\langle I, O\rangle$, namely $I \subseteq I'$, and $\mathcal{A}$ is (I'/O')-aware DBP. Since our "bad news results", namely the non-monotonicity and the NBW used in the lower bound proof in Theorem 5 are looping, they are valid also for (I/O)-aware and (I^+/O^-)-aware DBPness.

B On The Refinement Step in Section 7.2

Consider the example NBW $\mathcal{A}$ on the left of Figure 4, where the input signals are $\{i_1, i_2, i_3\}$, and the output signals are $\{o_1, o_2\}$. We assume that the signals are mutually exclusive, thus in each moment in time, exactly one input signal and exactly one output signal is on. Accordingly, there are six letters in the alphabet of $\mathcal{A}$, and it specifies the following behavior: as long as the environment generates i_1, the system has to respond with a prefix of $L = ((o_1 \cdot o_1)^+ \cdot (o_1 \cdot o_2)^+)^\omega$. Once the environment generates i_2 or i_3, all suffixes are correct. If the environment keeps generating i_1 forever, the system has to respond with a word in L. For simplicity, we illustrate the refinement only for the deterministic subset construction of $\mathcal{A}$.

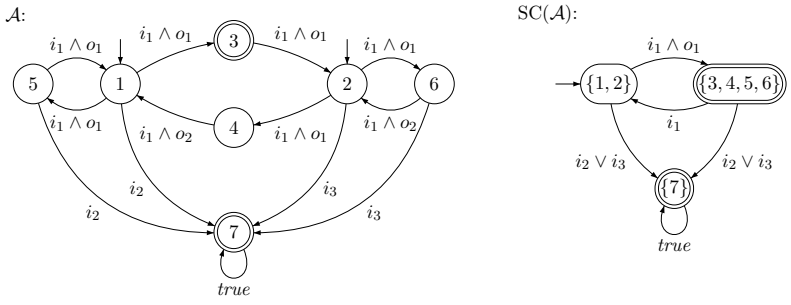

Fig. 4. An NBW $\mathcal{A}$ and its subset construction.

Refinement of states in NSC($\mathcal{A}$) that are associated with strict subsets is similar (and is necessary in order to filter out from the over-approximating automaton all the accepting runs on the counterexample word).

The NBW SC($\mathcal{A}$) on the right of Figure 4 is the subset construction applied to $\mathcal{A}$. For the input word ${i_1}^\omega$, a transducer that realizes $\mathcal{A}$ has to produce an output word in L above. However, $\tilde{\mathcal{A}}$ accepts also the words $\{i_1, o_1\}^\omega$ and $(\{i_1, o_1\} \cdot \{i_1, o_2\})^\omega$, which are not in $L(\mathcal{A})$. We show the refinement with respect to the word $w = (\{i_1, o_1\} \cdot \{i_1, o_2\})^\omega$.

The automaton $\mathcal{D}$ in the left of Figure 5 is a DBW for $\mathcal{A}$ (for the sake of readability, we omit from it an accepting sink $\{7\}$ and $(i_2 \vee i_3)$-transitions from all the states to the accepting sink). Every state in $\mathcal{D}$ consists of a subset of the states in $\mathcal{A}$ and additional information that describes which paths visited an accepting state recently (we follow the break-point construction used in [8], where the additional information is on states along paths that have not visited α – these are the underlined states). We refine the states along

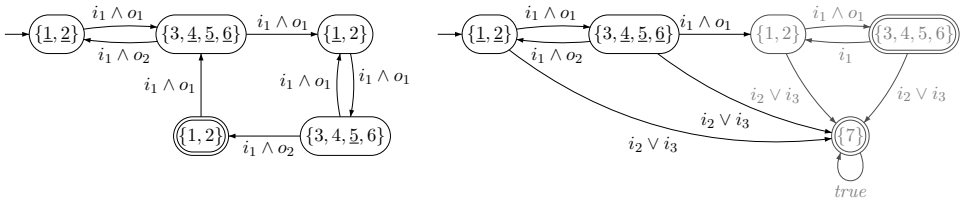

Fig. 5. A DBW $\mathcal{D}$ for $\mathcal{A}$ and SC($\mathcal{A}$) after refinement w.r.t. $(\{i_1, o_1\} \cdot \{i_1, o_2\})^\omega$.

the accepting run of SC($\mathcal{A}$) on w that are mapped to by states in the rejecting run of $\mathcal{D}$ on w, namely, both $\{1, 2\}$ and $\{3, 4, 5, 6\}$. The resulting NBW $\tilde{\mathcal{A}}$ is described on the right of Figure 5, where the states and edges that appear already in SC($\mathcal{A}$) are colored gray. Note that we do not have to construct $\mathcal{D}$, just generate, on-the-fly, its run on w. Note also that $\tilde{\mathcal{A}}$ is still an over-approximation of $\mathcal{A}$. However, it is not hard to see that an additional step of refinement, for example w.r.t. the word $\{i_1, o_1\} \cdot \{i_1, o_1\} \cdot w$, adds the copies of the state $\{1, 2\}$ with the necessary information, and thus is sufficient in order to get an NBW for which a winning strategy for Player OR realizes $\mathcal{A}$.

Finally, note that working with NSC($\mathcal{A}$), a winning strategy should use only states from which there is an i-transition for every $i \in \{i_1, i_2, i_3\}$, thus it has to start in $\{1, 2\}$, and move, given i_1, to a subset of $\{3, 4, 5, 6\}$ that is a superset of $\{5, 6\}$. Then, given i_1, it must go back to $\{1, 2\}$. Assuming that the strategy is memoryless, this induces a loop that is associated with a word not in $L(\mathcal{A})$. Thus, refinement is needed also in this case.

In many cases, refinement can be avoided. As an example, consider the NBW $\mathcal{B}$ over $2^{\{i,o\}}$ that appears on the left of Figure 6. The language of $\mathcal{B}$ is given by the LTL formula $GFi \iff GFo$. That is, $\mathcal{B}$ accepts every word that has infinitely many i-s iff it has infinitely many o-s. Note that $L(\mathcal{B})$ is not DBW-recognizable. The DBW in the right of Fig-

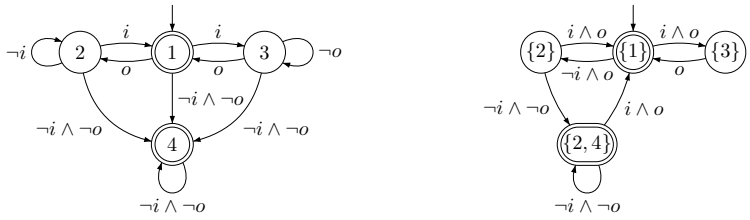

Fig. 6. An NBW $\mathcal{B}$ and a sub-automaton of NSC($\mathcal{B}$) that realizes $\mathcal{B}$.

ure 6 is a sub-automaton of NSC($\mathcal{B}$) on top of which player OR wins $G_{\text{SYN}}(\text{NSC}(\mathcal{B}), \{i\}, \{o\})$, and it can be viewed as a transducer that realizes $\mathcal{B}$. Thus, by over-approximating $\mathcal{B}$ and solving synthesis according to the heuristic that chooses a minimalistic winning strategy, we solve the synthesis problem for $\mathcal{B}$ avoiding determinization.

C Proofs

C.1 Proof of Lemma 1

Assume first that $\mathcal{A}$ is (I/O)-aware GFG, and let $g : (2^I)^* \to 2^O \times Q$ be a function that witnesses its (I/O)-aware GFGness. It is easy to see that the function $g_Q : (2^I)^* \to Q$ witnesses that $\mathcal{A}_{|I}$ is GFG. Indeed, the function g_Q produces accepting runs on all the words that are hopeful in $\mathcal{A}$, which are exactly the words in $L(\mathcal{A}_{|I})$. In addition, every run that is accepting in $\mathcal{A}$ is accepting in $\mathcal{A}_{|I}$.

For the other direction, assume that $\mathcal{A}_{|I}$ is GFG, and let $g' : (2^I)^* \to Q$ be a function that witnesses its GFGness. We describe a function $g : (2^I)^* \to 2^O \times Q$ that witnesses $\mathcal{A}$'s (I/O)-aware GFGness. First, we define $g_Q = g'$. We define g_O inductively as follows: $g_O(\epsilon) = \epsilon$, and for a word $w_I \cdot i \in (2^I)^* \times 2^I$, we have that $g_O(w_I \cdot i) = o$, where $o \in 2^O$ is such that $g_Q(w_I \cdot i) \in \delta(g_Q(w_I), i \cup o)$. Clearly, g is consistent with δ. In addition, note that for every $w_I \in (2^I)^\omega$, we have that $w_I \in L(\mathcal{A}_{|I})$ iff there exists $w_O \in (2^O)^\omega$ such that $w_I \oplus w_O \in L(\mathcal{A})$. It is not hard to see that in this case, $g_Q(\epsilon), g_Q(i_0), g_Q(i_0 \cdot i_1), \ldots$ is an accepting run on $w_I \oplus (g_O(\epsilon) \cdot g_O(i_0) \cdot g_O(i_0 \cdot i_1) \cdots)$. Thus, g witnesses $\mathcal{A}$'s (I/O)-aware GFGness, and we are done.

C.2 Proof of Lemma 2

Consider an NBW $\mathcal{A}$ over 2^X. For the first claim, note that he NBW $\mathcal{A}_{|\emptyset}$ is a single-letter NBW, thus its language is either empty, in which case it is clearly GFG, or it includes a single word, in which case its accepting run induces a function that witnesses its GFGness. Hence, $\mathcal{A}_{|\emptyset}$ is GFG, implying, by Lemma 1, that $\mathcal{A}$ is $(\emptyset/X)$-aware GFG. Also, since $\mathcal{A}_{|X} = \mathcal{A}$, the second claim follows from Lemma 1.

C.3 Proof of Theorem 2

We define $\mathcal{A}^k = \langle 2^{\{x_1, x_2 \ldots, x_{2k}\}}, Q, q_0, \delta, Q\rangle$, where $Q = \{q_0, q_1, \ldots, q_{k+1}\}$, and δ is defined, for all $1 \leq j \leq k$, as follows (see Figure 7 for an illustration of $\mathcal{A}^4$).

- $\delta(q_0, \{x_1, x_2, \ldots, x_{2j-1}\}) = q_j$,
- $\delta(q_0, \{x_1, x_2 \ldots, x_{2k}\}) = q_{k+1}$,
- $\delta(q_j, \{x_1, x_3, x_5, \ldots, x_{2(j-1)-1}\}) = q_j$, and
- $\delta(q_{k+1}, \{x_1, x_3, x_5, \ldots, x_{2k-1}\}) = q_{k+1}$.

Consider an index $1 \leq j \leq k$. Let $I = \{x_1, \ldots, x_{2j-1}\}$ and $O = \{x_{2j}, x_{2j+1}, \ldots, x_{2k}\}$. We show that $\mathcal{A}^k_{|I}$ is not GFG. Then, it follows from Lemma 1 that $\mathcal{A}^k$ is not (I/O)-aware GFG (see Figure 7 for illustrations of $\mathcal{A}^4_{|\{x_1,x_2,x_3\}}$ and $\mathcal{A}^4_{|\{x_1,x_2,x_3,x_4\}}$; in both, the states q_5 and q_4 are identical to q_3, and are omitted in the figure). Consider a function $g : (2^I)^* \to Q$. Note that $g(I) \in \{q_j, \ldots, q_{k+1}\}$. If $g(I) = q_j$, then g neglects the word $I \cdot$

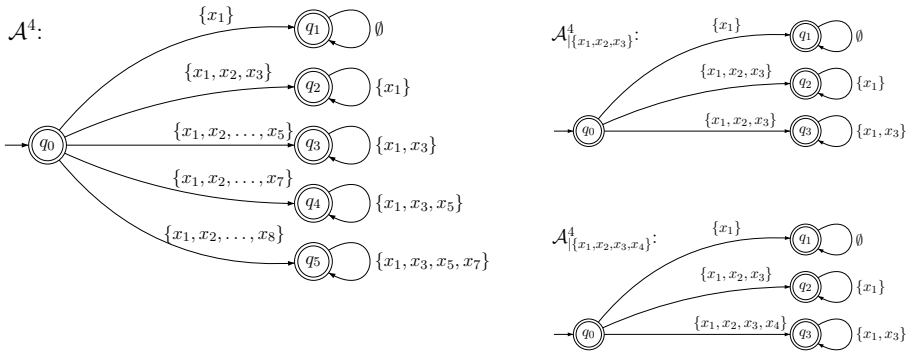

Fig. 7. The DLW $\mathcal{A}^4$ and its projections.

$(\{x_1, x_3, \ldots, x_{2j-1}\})^\omega \in L(\mathcal{A}^k_{|I})$. Otherwise, g neglects the word $I \cdot (\{x_1, x_3, \ldots, x_{2(j-1)-1}\})^\omega \in L(\mathcal{A}^k_{|I})$.

Now, let $I = \{x_1, \ldots, x_{2j}\}$ and $O = \{x_{2j+1}, \ldots, x_{2k}\}$. We prove that $\mathcal{A}^k$ is (I/O)-aware GFG by showing that $\mathcal{A}^k_{|I}$ is GFG. First, note that all the transitions in $\mathcal{A}^k_{|I}$ from q_0 to $q_1, \ldots, q_j$ are labeled with different strict subsets of I, and all the transitions in $\mathcal{A}^k_{|I}$ from q_0 to $q_{j+1}, \ldots, q_{k+1}$ are labeled with I. We show that a function $g : (2^I)^* \to Q$ such that $g(I) = q_{j+1}$ witnesses that $\mathcal{A}^k_{|I}$ is GFG. Indeed, since the self-loops in $q_{j+1}, q_{j+2}, \ldots, q_{k+1}$ are labeled with the same subset of I (namely, $\{x_1, x_3, x_5, \ldots, x_{2j-1}\}$), we have that g does not neglect any word in $L(\mathcal{A}^k_{|I})$.

C.4 Correctness of the reduction in the proof of Theorem 5

We prove that $\mathcal{A}$ is (I^+/O^-)-aware GFG iff φ is satisfiable.

Recall that $\mathcal{A}$ is (I^+/O^-)-aware GFG iff it is (I'/O')-aware GFG for partition $\langle I', O' \rangle$ that has less control than $\langle I, O \rangle$. Further, since $\mathcal{A}$ is deterministic in its transitions from q_0 and all those transitions are labeled with input letters, then $\mathcal{A}$ is (I'/O')-aware GFG iff all the NLWs $\mathcal{B}^1, \ldots, \mathcal{B}^n, \mathcal{C}^1, \ldots, \mathcal{C}^m$ are (I'/O')-aware GFG. Intuitively, the purpose of the NLWs $\mathcal{B}^1, \ldots, \mathcal{B}^n$ is to ensure that the choice of I' induces an assignment to X, by forcing I' to include, for every $x \in X$, at most one of x and $\tilde{x}$. The purpose of the NLWs $\mathcal{C}^1, \ldots, \mathcal{C}^m$ is to ensure that the assignment that I' induces satisfies φ.

We prove that for every $1 \leq j \leq n$, the NLW $\mathcal{B}^j$ is (I'/O')-aware GFG iff either $x_j \notin I'$ or $\tilde{x}_j \notin I'$. For the first direction, assume that $x_j \notin I'$. Then, a function $g : (2^{I'})^* \to 2^{O'} \times P_j$ such that $g_Q(\sigma) = p^j_1$ for all $\sigma \in 2^{I'}$ witnesses that $\mathcal{B}^j$ is (I'/O')-aware GFG, as all the words over $2^{I'}$ that are accepted from p^j_2 are accepted also from p^j_1. The case that $\tilde{x}_j \notin I'$ is symmetrical, with $g_Q(\sigma) = p^j_2$. For the other direction, we prove that if both $x_j \in I'$ and $\tilde{x}_j \in I'$, then $\mathcal{B}^j$ is not (I'/O')-aware GFG. First, if $g_Q(\sigma) = p^j_1$, then g neglects all the words $\sigma_1 \cdot \sigma_2^\omega \in (2^{I'})^\omega$, where $x_j, \tilde{x}_j \in \sigma_1$, $\tilde{x}_j \in \sigma_2$ and $x_j \notin \sigma_2$. Indeed, a run that follows g on these words gets stuck when it reads σ_2 from p^j_1. Note that all those words are in $L(\mathcal{B}^j_{|I'})$, as σ_2^ω can be accepted from p^j_2. The case that $g_Q(\sigma) = p^j_2$ is symmetrical, with the words $\sigma_1 \cdot \sigma_2^\omega$ such that $x_j, \tilde{x}_j \in \sigma_1$, $x_j \in \sigma_2$ and $\tilde{x}_j \notin \sigma_2$. Therefore,

all the automata $\mathcal{B}^1, \ldots, \mathcal{B}^n$ are (I'/O')-aware GFG iff I' includes, for every $x \in X$, at most one of $x, \tilde{x}$. We say that such I' is *legal*.

We continue and prove that for every $1 \leq j \leq m$, the NLW $\mathcal{C}^j$ is (I'/O')-aware GFG iff I' includes a signal that participates in $\tilde{c}_j$. Consider an index $1 \leq j \leq m$. In the first direction, if I' includes a signal that participates in $\tilde{c}_j$, then $\mathcal{C}^j_{|I'}$ is deterministic, and thus $\mathcal{C}^j$ is (I'/O')-aware GFG. For the other direction, assume by way of contradiction that all the signals that participate in $\tilde{c}_j$ are not in I', and that $\mathcal{C}^j$ is (I'/O')-aware GFG. Let $g : (2^{I'})^* \to 2^{O'} \times S_j$ be a function that witnesses $\mathcal{C}^j$'s (I'/O')-aware GFGness, and consider a letter $\sigma \in 2^{I'}$. If $g_Q(\sigma) = s^j_1$, then g neglects the word $\sigma \cdot \emptyset^\omega$. Otherwise, we have that $g_Q(\sigma) = s^j_2$, and g neglects the word $\sigma \cdot \{j\}^\omega$. Note that both $\sigma \cdot \emptyset^\omega$ and $\sigma \cdot \{j\}^\omega$ are in $L(\mathcal{C}^j_{|I'})$. We say that I' is *good* if it includes, for every j, at least one signal that participates in $\tilde{c}_j$. Note that all the automata $\mathcal{C}^1, \ldots, \mathcal{C}^m$ are (I'/O')-aware GFG iff I' is good. It follows that $\mathcal{A}$ is (I^+/O^-)-aware GFG iff there exists a partition $\langle I', O' \rangle$ that has less control than $\langle I, O \rangle$ such that I' is both legal and good. We show that the later holds iff φ is satisfiable.

In the first direction, let $\langle I', O' \rangle$ be a partition of $I \cup O$ that has less control than $\langle I, O \rangle$ and such that I' is legal and good. We define an assignment $f : X \to \{\mathit{true}, \mathit{false}\}$, where $f(x) = \mathit{true}$ iff $x \in I'$. Note that I' is legal, thus f is well-defined. In addition, since I' is good, then f satisfies φ. For the other direction, let $f : X \to \{\mathit{true}, \mathit{false}\}$ be an assignment that satisfies φ. We define $I' = I \cup \{x : f(x) = \mathit{true}\} \cup \{\tilde{x} : f(x) = \mathit{false}\}$. It is easy to see that I' is both legal and good.

C.5 Proof of Lemma 3

Consider the LTL formula $\psi_n = (G\neg a) \vee F(a \wedge X^n G b)$ over $\{a, b\}$. Note that ψ_n has the flavor of the regular language $L_n = (0{+}1)^* \cdot 0 \cdot (0{+}1)^{n-1}$ over $\{0, 1\}$. The language L_n is a classical example to the succinctness of nondeterministic automata: a nondeterministic automaton for L_n can guess whether each 0 is located in the desired position, namely n letters before the end of the word, and needs only $O(n)$ states. On the other hand, a deterministic and even a GFG automaton for L_n needs to remember the last n letters and needs at least 2^n states. Indeed, a GFG strategy cannot map different words of length n to the same state, as a suffix that inserts only one of them to L_n would fool the automaton. Back to ψ_n, where the property Gb plays the role of "end of word" in L_n. Here too, a nondeterministic automaton $\mathcal{A}_n$ for ψ_n may guess whether each occurrence of a appears n positions before a tail of b's starts; and here too, if we do not distinguish between input and output signals, then a GFG automaton for ψ_n is exponential in n. On the other hand, if $I = \{a\}$ and $O = \{b\}$, then an (I/O)-aware GFG automaton $\mathcal{A}'_n$ can assume a cooperative behavior of the system in which b is always valid, have a single state, and still (I/O)-cover $\mathcal{A}_n$.