Minimizing GFG Transition-Based Automata

Bader Abu Radi and Orna Kupferman
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel
bader.aburadi@gmail.com, orna@cs.huji.ac.il

Abstract

While many applications of automata in formal methods can use nondeterministic automata, some applications, most notably synthesis, need deterministic or *good-for-games* automata. The latter are nondeterministic automata that can resolve their nondeterministic choices in a way that only depends on the past. The *minimization* problem for nondeterministic and deterministic Büchi and co-Büchi word automata are PSPACE-complete and NP-complete, respectively. We describe a polynomial minimization algorithm for good-for-games *co-Büchi* word automata with *transition-based* acceptance. Thus, a run is accepting if it traverses a set of designated transitions only finitely often. Our algorithm is based on a sequence of transformations we apply to the automaton, on top of which a minimal quotient automaton is defined.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Automata over infinite objects

Keywords and phrases Minimization, Deterministic co-Büchi Automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.95

1 Introduction

Automata theory is one of the longest established areas in Computer Science. A classical problem in automata theory is *minimization*: generation of an equivalent automaton with a minimal number of states. For automata on finite words, the picture is well understood: For nondeterministic automata, minimization is PSPACE-complete [16], whereas for deterministic automata, a minimization algorithm, based on the Myhill-Nerode right congruence [28, 29], generates in polynomial time a canonical minimal deterministic automaton [14]. Essentially, the canonical automaton, a.k.a. the *quotient automaton*, is obtained by merging equivalent states.

A prime application of automata theory is specification, verification, and synthesis of reactive systems [8, 36]. The automata-theoretic approach considers relationships between systems and their specifications as relationships between languages. Since we care about the on-going behavior of nonterminating systems, the automata run on infinite words. Acceptance in such automata is determined according to the set of states that are visited infinitely often along the run. In Büchi automata [5] (NBW and DBW, for nondeterministic and deterministic Büchi word automata, respectively), the acceptance condition is a subset α of states, and a run is accepting iff it visits α infinitely often. Dually, in co-Büchi automata (NCW and DCW), a run is accepting iff it visits α only finitely often. In spite of the extensive use of automata on infinite words in verification and synthesis algorithms and tools, some fundamental problems around their minimization are still open. For nondeterministic automata, minimization is PSPACE-complete, as it is for automata on finite words. Before we describe the situation for deterministic automata, let us elaborate some more on the power of nondeterminism in the context of automata on infinite words, as this would be relevant to our contribution.

For automata on finite words, nondeterminism does not increase the expressive power, yet it leads to an exponential succinctness [31]. For automata on infinite words, nondeterminism may increase the expressive power and also leads to an exponential succinctness. For example, NBWs are strictly more expressive than DBWs [21]. In some applications of automata on infinite words, such as model checking, algorithms can proceed with nondeterministic automata, whereas in other applications, such as synthesis and control, they cannot. There, the advantages of nondeterminism are lost, and the algorithms involve complicated determinization constructions [32] or acrobatics for circumventing

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 95; pp. 95:1–95:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

determinization [20]. 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.

The study of nondeterministic automata that can resolve their nondeterministic choices in a way that only depends on the past and still accept all words in the language started already in 1996 [19], where the setting is modeled by means of tree automata for derived languages. It then continued by means of *good for games* (GFG) automata, introduced in [13].[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* (DBP); that is, ones that just add transitions on top of a deterministic automaton. In fact, the GFG automata constructed in [13] are DBP.[2]

In terms of expressive power, it is shown in [19, 30] that GFG automata with an acceptance condition γ (e.g., Büchi) are as expressive as deterministic γ automata. The picture in terms of succinctness is diverse. For automata on finite words, GFG automata are always DBP [19, 26]. For automata on infinite words, in particular NBWs and NCWs, GFG automata need not be DBP [3]. Moreover, the best known determinization construction for GFG-NBWs is quadratic, whereas determinization of GFG-NCWs has an exponential blow-up lower bound [17]. Thus, in terms of succinctness, GFG automata on infinite words are more succinct (possibly even exponentially) than deterministic ones. Further research studies characterization, typeness, complementation, and further constructions and decision procedures for GFG automata [2, 4, 17].

Back to the minimization problem. Recall that for finite words, an equivalent minimal deterministic automaton can be obtained by merging equivalent states. A similar algorithm is valid for deterministic *weak* automata on infinite words: DBWs in which each strongly connected component is either contained in α or is disjoint form α [23, 27]. For general DBWs (and hence, also DCWs, as the two dualize each other), merging of equivalent states fails, and minimization is NP-complete [33].

The intractability of the minimization problem has led to a development of numerous heuristics. The heuristics either relax the minimality requirement, for example algorithms based on *fair bisimulation* [10], which reduce the state space but need not return a minimal automaton, or relax the equivalence requirement, for example algorithms based on *hyper-minimization* [1, 15] or *almost-equivalence* [33], which come with a guarantee about the difference between the language of the original automaton and the ones generated by the algorithm. In some cases, these algorithms do generate of a minimal equivalent automaton (in particular, applying relative minimization based on almost equivalence on a deterministic weak automaton results in an equivalent minimal weak automaton [33]), but in general, they are only heuristics. In an orthogonal line of work, researchers have studied minimization in richer settings of automata on finite words. One direction is to allow some nondeterminism. As it turns out, however, even the slightest extension of the deterministic model towards a nondeterministic one, for example by allowing at most one nondeterministic choice in every accepting computation or allowing just two initial states instead of one, results in NP-complete minimization problems [24]. Another direction is a study of quantitative settings. Here, the picture is diverse. For example, minimization of deterministic lattice automata [18] is polynomial for automata over linear lattices and is NP-complete for general lattices [11], and minimization of deterministic

[1] GFGness is also used in [6] in the framework of cost functions under the name "history-determinism".

[2] As explained in [13], the fact that the GFG automata constructed there are DBP does not contradict their usefulness in practice, as their transition relation is simpler than the one of the embodied deterministic automaton and it can be defined symbolically.

weighted automata over the tropical semiring is polynomial [25], yet the problem is open for general semirings.

Proving NP-hardness for DBW minimization, Schewe used a reduction from the vertex-cover problem [33]. Essentially[3], given a graph $G = \langle V, E \rangle$, we seek a minimal DBW for the language L_G of words of the form $v_{i_1}^+ \cdot v_{i_2}^+ \cdot v_{i_3}^+ \cdots \in V^\omega$, where for all $j \geq 1$, we have that $\langle v_{i_j}, v_{i_{j+1}} \rangle \in E$. We can recognize L_G by an automaton obtained from G by adding self loops to all vertices, labelling each edge by its destination, and requiring a run to traverse infinitely many original edges of G. Indeed, such runs correspond to words that traverse an infinite path in G, possibly looping at vertices, but not getting trapped in a self loop, as required by L_G. When, however, the acceptance condition is defined by a set of vertices, rather than edges, we need to duplicate some states, and a minimal duplication corresponds to a minimal vertex cover. Thus, a natural question arises: Is there a polynomial minimization algorithms for DBWs and DCWs whose acceptance condition is *transition based*? Beyond the theoretical interest, there is recently growing use of transition-based automata in practical applications, with evidences they offer a simpler translation of LTL formulas to automata and enable simpler constructions and decision procedures [7,9,22,34].

In this paper we present a significant step towards a positive answer to this question and describe a polynomial-time algorithm for the minimization of GFG transition-based NCWs. Consider a GFG-NCW $\mathcal{A}$. Our algorithm is based on a chain of transformations we apply to $\mathcal{A}$. Some of the transformations are introduced in [17], in algorithms for deciding GFGness. We add two more transformations and prove that they guarantee minimality. Our reasoning is based on a careful analysis of the *safe components* of $\mathcal{A}$, namely the components obtained by removing transitions in α. We show that a minimal GFG-NCW equivalent to $\mathcal{A}$ can be obtained by defining an order on the safe components, and applying the quotient construction on a GFG-NCW obtained by restricting attention to states that belong to components that form a frontier in this order.

The paper is organized as follows. In Section 2, we define GFG-NCWs and some properties of GFG-NCWs that can be attained in polynomial time using existing results. In Section 3, we describe two additional properties and prove that they guarantee minimality. Then, in Sections 4 – 5, we show how the two properties can be attained in polynomial time, thus concluding our minimization procedure. In Section 6, we discuss how our results contribute to the quest for efficient DBW and DCW minimization.

2 Preliminaries

For a finite nonempty alphabet Σ, an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ. A *language* $L \subseteq \Sigma^\omega$ is a set of words. We denote the empty word by ϵ, the set of finite words over Σ by Σ^*. For $i \geq 0$, we use $w[1, i]$ to denote the (possibly empty) prefix $\sigma_1 \cdot \sigma_2 \cdots \sigma_i$ of w and use $w[i + 1, \infty]$ to denote its suffix $\sigma_{i+1} \cdot \sigma_{i+2} \cdots$.

A *nondeterministic automaton* over infinite words is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, $\delta : Q \times \Sigma \to 2^Q \setminus \emptyset$ is a *transition function*, and α is an *acceptance condition*, to be defined below. For states q and s and a letter $\sigma \in \Sigma$, we say that s is a σ-successor of q if $s \in \delta(q, \sigma)$. The *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, is defined as its number of states, thus, $|\mathcal{A}| = |Q|$. Note that $\mathcal{A}$ is *total*, in the sense that it has at least one successor for each state and letter, and that $\mathcal{A}$ may be *nondeterministic*, as the transition function may specify several successors for each state and letter. If $|\delta(q, \sigma)| = 1$ for every state $q \in Q$ and letter $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*.

[3] The exact reduction is more complicated and involves an additional letter that is required for cases in which vertices in the graph have similar neighbours.

When $\mathcal{A}$ runs on an input word, it starts in the initial state and proceeds according to the transition function. Formally, a *run* of $\mathcal{A}$ on $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states $r = r_0, r_1, r_2, \ldots \in Q^\omega$, such that $r_0 = q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$. We sometimes extend δ to sets of states and finite words. Then, $\delta : 2^Q \times \Sigma^* \to 2^Q$ is such that for every $S \in 2^Q$, finite word $u \in \Sigma^*$, and letter $\sigma \in \Sigma$, we have that $\delta(S, \epsilon) = S$, $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$, and $\delta(S, u \cdot \sigma) = \delta(\delta(S, u), \sigma)$. Thus, $\delta(S, u)$ is the set of states that $\mathcal{A}$ may reach when it reads u from some state in S.

The transition function δ induces a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, where for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, we have that $\langle q, \sigma, s \rangle \in \Delta$ iff $s \in \delta(q, \sigma)$. We sometimes view the run $r = r_0, r_1, r_2, \ldots$ on $w = \sigma_1 \cdot \sigma_2 \cdots$ as an infinite sequence of successive transitions $\langle r_0, \sigma_1, r_1 \rangle, \langle r_1, \sigma_2, r_2 \rangle, \ldots \in \Delta^\omega$. The acceptance condition α determines which runs are "good". We consider here *transition-based* automata, in which α refers to the set of transitions that are traversed infinitely often during the run; specifically, $\alpha \subseteq \Delta$. We use the terms *α-transitions* and *$\bar{\alpha}$-transitions* to refer to transitions in α and in $\Delta \setminus \alpha$, respectively. We also refer to restrictions δ^α and $\delta^{\bar{\alpha}}$ of δ, where for all $q, s \in Q$ and $\sigma \in \Sigma$, we have that $s \in \delta^\alpha(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \alpha$, and $s \in \delta^{\bar{\alpha}}(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \Delta \setminus \alpha$. For a run $r \in \Delta^\omega$, let $\mathit{inf}(r) \subseteq \Delta$ be the set of transitions that r traverses infinitely often. Thus, $\mathit{inf}(r) = \{\langle q, \sigma, s \rangle \in \Delta : q = r_i, \sigma = \sigma_{i+1}$ and $s = r_{i+1}$ for infinitely many i's$\}$. In *co-Büchi* automata, a run r is *accepting* iff $\mathit{inf}(r) \cap \alpha = \emptyset$, thus if r traverses transitions in α only finitely often. A run that is not accepting is *rejecting*. A word w is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. Two automata are *equivalent* if their languages are equivalent. We use tNCW and tDCW to abbreviate nondeterministic and deterministic transition-based co-Büchi automata over infinite words, respectively.

For a state $q \in Q$ of an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, we define $\mathcal{A}^q$ to be the automaton obtained from $\mathcal{A}$ by setting the initial state to be q. Thus, $\mathcal{A}^q = \langle \Sigma, Q, q, \delta, \alpha \rangle$. We say that two states $q, s \in Q$ are *equivalent*, denoted $q \sim_\mathcal{A} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. The automaton $\mathcal{A}$ is *semantically deterministic* if different nondeterministic choices lead to equivalent states. Thus, for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ-successors of q are equivalent: for every two states $s, s' \in Q$ such that $\langle q, \sigma, s \rangle$ and $\langle q, \sigma, s' \rangle$ are in Δ, we have that $s \sim_\mathcal{A} s'$. The following proposition follows immediately from the definitions.

► **Proposition 1.** *Consider a semantically deterministic automaton $\mathcal{A}$, states $q, s \in Q$, letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, q' \rangle, \langle s, \sigma, s' \rangle \in \Delta$. If $q \sim_\mathcal{A} s$, then $q' \sim_\mathcal{A} s'$.*

A tNCW $\mathcal{A}$ is *safe deterministic* if by removing its α-transitions, we get a (possibly not total) deterministic automaton. Thus, $\mathcal{A}$ is *safe deterministic* if for every state $q \in Q$ and letter $\sigma \in \Sigma$, it holds that $|\delta^{\bar{\alpha}}(q, \sigma)| \leq 1$. We refer to the components we get by removing $\mathcal{A}$'s α-transitions as the *safe components* of $\mathcal{A}$, and we denote the set of safe components of $\mathcal{A}$ by $S(\mathcal{A})$. For a safe component $S \in S(\mathcal{A})$, the *size* of S, denoted $|S|$, is the number of states in S. Note that an accepting run of $\mathcal{A}$ eventually gets trapped in one of $\mathcal{A}$'s safe components.

An automaton $\mathcal{A}$ is *good for games* (*GFG*, for short) if its nondeterminism can be resolved based on the past, thus on the prefix of the input word read so far. Formally, $\mathcal{A}$ is *GFG* if there exists a *strategy* $f : \Sigma^* \to Q$ such that the following holds:

1. The strategy f is consistent with the transition function. That is, for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $\langle f(u), \sigma, f(u \cdot \sigma) \rangle \in \Delta$.
2. Following f causes $\mathcal{A}$ to accept all the words in its language. That is, for every infinite word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$, if $w \in L(\mathcal{A})$, then the run $f(w[1, 0]), f(w[1, 1]), f(w[1, 2]), \ldots$, which we denote by $f(w)$, is accepting.

We say that the strategy f *witnesses* $\mathcal{A}$'s GFGness. For an automaton $\mathcal{A}$, we say that a state q of $\mathcal{A}$ is *GFG* if $\mathcal{A}^q$ is GFG. Finally, we say that a GFG-tNCW $\mathcal{A}$ is *minimal* if for every equivalent GFG-tNCW $\mathcal{B}$, it holds that $|\mathcal{A}| \leq |\mathcal{B}|$.

Consider a directed graph $G = \langle V, E\rangle$. A *strongly connected set* in G (SCS, for short) is a set $C \subseteq V$ such that for every two vertices $v, v' \in C$, there is a path from v to v'. A SCS is *maximal* if it is maximal w.r.t containment, that is, for every non-empty set $C' \subseteq V \setminus C$, it holds that $C \cup C'$ is not a SCS. The *maximal strongly connected sets* are also termed *strongly connected components* (SCCs, for short). The *SCC graph of* G is the graph defined over the SCCs of G, where there is an edge from a SCC C to another SCC C' iff there are two vertices $v \in C$ and $v' \in C'$ with $\langle v, v'\rangle \in E$. A SCC is *ergodic* iff it has no outgoing edges in the SCC graph. The SCC graph of G can be computed in linear time by standard SCC algorithms [35]. An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$ induces a directed graph $G_{\mathcal{A}} = \langle Q, E\rangle$, where $\langle q, q'\rangle \in E$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q'\rangle \in \Delta$. The SCSs and SCCs of $\mathcal{A}$ are those of $G_{\mathcal{A}}$. We say that a tNCW $\mathcal{A}$ is *normal* if all the safe components of $\mathcal{A}$ are SCSs. That is, for all states q and s of $\mathcal{A}$, if there is a path of $\bar{\alpha}$-transition from q to s, then there is also a path of $\bar{\alpha}$-transition from s to q.

We now combine several properties defined above and say that a GFG-tNCW $\mathcal{A}$ is *nice* if all the states in $\mathcal{A}$ are reachable and GFG, and $\mathcal{A}$ is normal, safe deterministic, and semantically deterministic. In the theorem below we combine arguments from [17] showing that each of these properties can be obtained in at most polynomial time, and without the properties being conflicting. For some properties, we give an alternative and simpler proof.

▶ **Theorem 2.** [17] *Every GFG-tNCW $\mathcal{A}$ can be turned, in polynomial time, into an equivalent nice GFG-tNCW $\mathcal{B}$ such that $|\mathcal{B}| \leq |\mathcal{A}|$.*

Proof. It is shown in [17] that one can decide the GFGness of a tNCW $\mathcal{A}$ in polynomial time. The proof goes through an intermediate step where the authors construct a two-players game such that if the first player does not win the game, then $\mathcal{A}$ is not GFG, and otherwise a winning strategy for him induces a safe-deterministic GFG-tNCW $\mathcal{B}$ equivalent to $\mathcal{A}$. As we start with a GFG-tNCW $\mathcal{A}$, such a winning strategy is guaranteed to exist, and we obtain an equivalent safe-deterministic GFG-tNCW $\mathcal{B}$ in polynomial time. In fact, it can be shown that $\mathcal{B}$ is also semantically deterministic. Yet, for completeness we give below a general procedure for semantic determinization.

For a tNCW $\mathcal{A}$, we say that a transition $\langle q, \sigma, s\rangle \in \Delta$ is *covering* if for every transition $\langle q, \sigma, s'\rangle$, it holds that $L(\mathcal{A}^{s'}) \subseteq L(\mathcal{A}^s)$. If $\mathcal{A}$ is GFG and f is a strategy witnessing its GFGness, we say that a state q of $\mathcal{A}$ is *used by* f if there is a finite word u with $f(u) = q$, and we say that a transition $\langle q, \sigma, q'\rangle$ of $\mathcal{A}$ is *used by* f if there is a finite word u with $f(u) = q$ and $f(u\sigma) = q'$. Since states that are not GFG can be detected in polynomial time, and as all states that are used by a strategy that witnesses $\mathcal{B}$'s GFGness are GFG, the removal of non-GFG states does not affect $\mathcal{B}$'s language. Note that removing the non-GFG states may result in a non-total automaton, in which case we add a rejecting sink. Now, using the fact that language containment of GFG-tNCWs can be checked in polynomial time [12, 17], and transitions that are used by strategies are covering [17], one can semantically determinize $\mathcal{B}$ by removing non-covering transitions.

States that are not reachable are easy to detect, and their removal does not affect $\mathcal{B}$'s language. Normalization is also easy to obtain and involves adding some existing transitions to α [17]. Indeed, if the safe components of $\mathcal{B}$ are not SCSs, then every $\bar{\alpha}$-transition connecting different SCCs of $\mathcal{B}$'s safe components can be added to α without affecting the acceptance of runs in $\mathcal{B}$, as every accepting run traverses such transitions only finitely often. Thus, the language and GFGness of all states are not affected. Finally, it is not hard to verify that the properties, in the order we obtain them in the proof, are not conflicting, and thus the described sequence of transformations results in a nice GFG-tNCW. ◀

3 A Sufficient Condition for GFG-tNCW Minimality

In this section, we define two additional properties for nice GFG-tNCWs, namely *safe-centralized* and *safe-minimal*, and we prove that nice GFG-tNCWs that attain these properties are minimal. In Sections 4 – 5, we are going to show that the two properties can be attained in polynomial time. Before we start, let us note that a GFG-tNCW may be nice and still not be minimal. A simple example is a GFG-tNCW $\mathcal{A}_{\mathsf{fm}}$ for the language $(a+b)^* \cdot a^\omega$ that has two states, both with a $\bar{\alpha}$-self-loop labeled a and an α-transition labeled b to the other state. It is easy to see that $\mathcal{A}_{\mathsf{fm}}$ is nice but not minimal.

Consider a tNCW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. A run r of $\mathcal{A}$ is *safe* if it does not traverse α-transitions. The *safe language* of $\mathcal{A}$, denoted $L_{safe}(\mathcal{A})$, is the set of infinite words w, such that there is a safe run of $\mathcal{A}$ on w. Recall that two states $q, s \in Q$ are equivalent ($q \sim_{\mathcal{A}} s$) if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. Then, q and s are *strongly-equivalent*, denoted $q \approx_{\mathcal{A}} s$, if $q \sim_{\mathcal{A}} s$ and $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{A}^s)$. Finally, q is *subsafe-equivalent to* s, denoted $q \precsim_{\mathcal{A}} s$, if $q \sim_{\mathcal{A}} s$ and $L_{safe}(\mathcal{A}^q) \subseteq L_{safe}(\mathcal{A}^s)$. Note that the three relations are transitive. When $\mathcal{A}$ is clear from the context, we omit it from the notations, thus write $L_{safe}(q)$, $q \precsim s$, etc. The tNCW $\mathcal{A}$ is *safe-minimal* if it has no strongly-equivalent states. Then, $\mathcal{A}$ is *safe-centralized* if for every two states $q, s \in Q$, if $q \precsim s$, then q and s are in the same safe component of $\mathcal{A}$.

▶ **Example 3.** The nice GFG-tNCW $\mathcal{A}_{\mathsf{fm}}$ described above is neither safe-minimal (its two states are strongly-equivalent) nor safe-centralized (its two states are in different safe components). As another example, consider the tDCW $\mathcal{A}$ appearing in Figure 1. The dashed transitions are α-transitions. All the states of $\mathcal{A}$ are equivalent, yet they all differ in their safe language. Accordingly, $\mathcal{A}$ is safe-minimal. Since $a^\omega = L_{safe}(\mathcal{A}^{q_2}) \subseteq L_{safe}(\mathcal{A}^{q_0})$, we have that $q_2 \precsim q_0$. Hence, as q_0 and q_2 are in different safe components, the tDCW $\mathcal{A}$ is not safe-centralized.

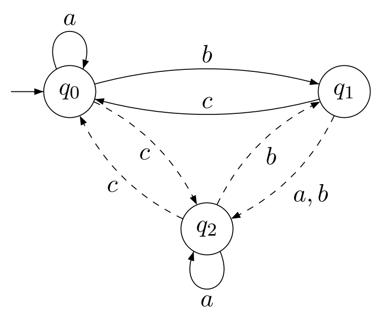

Figure 1 The tDCW $\mathcal{A}$.

▶ **Proposition 4.** *Consider a nice GFG-tNCW $\mathcal{A}$ and states q and s of $\mathcal{A}$ such that $q \approx s$ ($q \precsim s$). For every letter $\sigma \in \Sigma$ and $\bar{\alpha}$-transition $\langle q, \sigma, q' \rangle$, there is an $\bar{\alpha}$-transition $\langle s, \sigma, s' \rangle$ such that $q' \approx s'$ ($q' \precsim s'$, respectively).*

Proof. We prove the proposition for the case $q \approx s$. The case $q \precsim s$ is similar. Since $\mathcal{A}$ is normal, the existence of the $\bar{\alpha}$-transition $\langle q, \sigma, q' \rangle$ implies that there is a safe run from q' back to q. Hence, there is a word $z \in L_{safe}(\mathcal{A}^{q'})$. Clearly, $\sigma \cdot z$ is in $L_{safe}(\mathcal{A}^q)$. Now, since $q \approx s$, we have that $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{A}^s)$. In particular, $\sigma \cdot z \in L_{safe}(\mathcal{A}^s)$, and thus there is a $\bar{\alpha}$-transition $\langle s, \sigma, s' \rangle$. We prove that $q' \approx s'$. Since $L(\mathcal{A}^q) = L(\mathcal{A}^s)$ and $\mathcal{A}$ is semantically deterministic, then, by Proposition 1, we have that $L(\mathcal{A}^{q'}) = L(\mathcal{A}^{s'})$. It is left to prove that $L_{safe}(\mathcal{A}^{q'}) = L_{safe}(\mathcal{A}^{s'})$. We prove that $L_{safe}(\mathcal{A}^{q'}) \subseteq L_{safe}(\mathcal{A}^{s'})$. The second direction is similar. Since $\mathcal{A}$ is safe deterministic, the transition $\langle s, \sigma, s' \rangle$ is the only σ-labeled $\bar{\alpha}$-transition from s. Hence, if by contradiction there is a word $z \in L_{safe}(\mathcal{A}^{q'}) \setminus L_{safe}(\mathcal{A}^{s'})$, we get that $\sigma \cdot z \in L_{safe}(\mathcal{A}^q) \setminus L_{safe}(\mathcal{A}^s)$, contradicting the fact that $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{A}^s)$. ◀

We continue with propositions that relate two automata, $\mathcal{A} = \langle \Sigma, Q_{\mathcal{A}}, q^0_{\mathcal{A}}, \delta_{\mathcal{A}}, \alpha_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle \Sigma, Q_{\mathcal{B}}, q^0_{\mathcal{B}}, \delta_{\mathcal{B}}, \alpha_{\mathcal{B}} \rangle$. We assume that $Q_{\mathcal{A}}$ and $Q_{\mathcal{B}}$ are disjoint, and extend the $\sim$, $\approx$, and $\precsim$ relations to states in $Q_{\mathcal{A}} \cup Q_{\mathcal{B}}$ in the expected way. For example, for $q \in Q_{\mathcal{A}}$ and $s \in Q_{\mathcal{B}}$, we use $q \sim s$ to indicate that $L(\mathcal{A}^q) = L(\mathcal{B}^s)$.

▶ **Proposition 5.** *Let $\mathcal{A}$ and $\mathcal{B}$ be equivalent nice GFG-tNCWs. For every state $q \in Q_{\mathcal{A}}$, there is a state $s \in Q_{\mathcal{B}}$ such that $q \precsim s$.*

Proof. Let g be a strategy witnessing $\mathcal{B}$'s GFGness. Consider a state $q \in Q_\mathcal{A}$. Let $u \in \Sigma^*$ be such that $q \in \delta_\mathcal{A}(q^0_\mathcal{A}, u)$. Since $\mathcal{A}$ and $\mathcal{B}$ are equivalent and semantically deterministic, an iterative application of Proposition 1 implies that for every state $q' \in \delta_\mathcal{B}(q^0_\mathcal{B}, u)$, we have $q \sim q'$. In particular, $q \sim g(u)$. If $L_{safe}(\mathcal{A}^q) = \emptyset$, then we are done, as $L_{safe}(\mathcal{A}^q) \subseteq L_{safe}(\mathcal{B}^{g(u)})$. If $L_{safe}(\mathcal{A}^q) \neq \emptyset$, then the proof proceeds as follows. Assume by way of contradiction that for every state $s \in Q_\mathcal{B}$ that is equivalent to q, it holds that $L_{safe}(\mathcal{A}^q) \not\subseteq L_{safe}(\mathcal{B}^s)$. We define an infinite word z such that $\mathcal{A}$ accepts $u \cdot z$, yet $g(u \cdot z)$ is a rejecting run of $\mathcal{B}$. Since $\mathcal{A}$ and $\mathcal{B}$ are equivalent, this contradicts the fact that g witnesses $\mathcal{B}$'s GFGness.

We define z as follows. Let $s_0 = g(u)$. Since $L_{safe}(\mathcal{A}^q) \not\subseteq L_{safe}(\mathcal{B}^{s_0})$, there is a finite nonempty word z_1 such that there is a safe run of $\mathcal{A}^q$ on z_1, but every run of $\mathcal{B}^{s_0}$ on z_1 is not safe. In particular, the run of $\mathcal{B}^{s_0}$ that is induced by g, namely $g(u), g(u \cdot z_1[1,1]), g(u \cdot z_1[1,2]), \ldots, g(u \cdot z_1)$, traverses an α-transition. Since $\mathcal{A}$ is normal, we can define z_1 so the safe run of $\mathcal{A}^q$ on z_1 ends in q. Let $s_1 = g(u \cdot z_1)$. We have so far two finite runs: $q \xrightarrow{z_1} q$ and $s_0 \xrightarrow{z_1} s_1$, where the first run is safe, and the second is not. Now, since $q \sim s_0$, then again by Proposition 1 we have that $q \sim s_1$, and by applying the same considerations, we can define a finite nonempty word z_2 and $s_2 = g(u \cdot z_1 \cdot z_2)$ such that $q \xrightarrow{z_2} q$ and $s_1 \xrightarrow{z_2} s_2$, where the first run is safe, and the second is not. After at most $|Q_\mathcal{B}|$ iterations, we get that there are $0 \leq j_1 < j_2 \leq |Q_\mathcal{B}|$ such that $s_{j_1} = s_{j_2}$, and define $z = z_1 \cdots z_2 \cdots z_{j_1} \cdot (z_{j_1+1} \cdots z_{j_2})^\omega$. Since $j_1 < j_2$, the extension $z_{j_1+1} \cdots z_{j_2}$ is nonempty and thus z is infinite. On the one hand, since $q \in \delta_\mathcal{A}(q^0_\mathcal{A}, u)$ and there is a safe run of $\mathcal{A}^q$ on z, we have that $u \cdot z \in L(\mathcal{A})$. On the other hand, the run $g(u \cdot z)$ traverses an α-transitions infinitely often, and is thus rejecting. ◀

▶ **Proposition 6.** *Let $\mathcal{A}$ and $\mathcal{B}$ be equivalent nice GFG-tNCWs. For every state $p \in Q_\mathcal{A}$, there are states $q \in Q_\mathcal{A}$ and $s \in Q_\mathcal{B}$ such that $p \precsim q$ and $q \approx s$.*

Proof. The proposition follows from the combination of Proposition 5 with the transitivity of $\precsim$ and the fact $Q_\mathcal{A}$ and $Q_\mathcal{B}$ are finite. Formally, consider the directed bipartite graph $G = \langle Q_\mathcal{A} \cup Q_\mathcal{B}, E \rangle$, where $E \subseteq (Q_\mathcal{A} \times Q_\mathcal{B}) \cup (Q_\mathcal{B} \times Q_\mathcal{A})$ is such that $\langle p_1, p_2 \rangle \in E$ iff $p_1 \precsim p_2$. Proposition 5 implies that E is total. That is, from every state in $Q_\mathcal{A}$ there is an edge to some state in $Q_\mathcal{B}$, and from every state in $Q_\mathcal{B}$ there is an edge to some state in $Q_\mathcal{A}$. Since $Q_\mathcal{A}$ and $Q_\mathcal{B}$ are finite, this implies that for every $p \in Q_\mathcal{A}$, there is a path in G that starts in p and reaches a state $q \in Q_\mathcal{A}$ (possibly $q = p$) that belongs to a nonempty cycle. We take s to be some state in $Q_\mathcal{B}$ in this cycle. By the transitivity of $\precsim$, we have that $p \precsim q$, $q \precsim s$, and $s \precsim q$. The last two imply that $q \approx s$, and we are done. ◀

▶ **Lemma 7.** *Consider a nice GFG-tNCW $\mathcal{A}$. If $\mathcal{A}$ is safe-centralized and safe-minimal, then for every nice GFG-tNCW $\mathcal{B}$ equivalent to $\mathcal{A}$, there is an injection $\eta : S(\mathcal{A}) \to S(\mathcal{B})$ such that for every safe component $T \in S(\mathcal{A})$, it holds that $|T| \leq |\eta(T)|$.*

Proof. We define η as follows. Consider a safe component $T \in S(\mathcal{A})$. Let p_T be some state in T. By Proposition 6, there are states $q_T \in Q_\mathcal{A}$ and $s_T \in Q_\mathcal{B}$ such that $p_T \precsim q_T$ and $q_T \approx s_T$. Since $\mathcal{A}$ is safe-centralized, the states p_T and q_T are in the same safe component, thus $q_T \in T$. We define $\eta(T)$ to be the safe component of s_T in $\mathcal{B}$. We show that η is an injection; that is, for every two safe components T_1 and T_2 in $S(\mathcal{A})$, it holds that $\eta(T_1) \neq \eta(T_2)$. Assume by way of contradiction that T_1 and T_2 are such that s_{T_1} and s_{T_2}, chosen as described above, are in the same safe component of $\mathcal{B}$. Then, there is a safe run from s_{T_1} to s_{T_2}. Since $s_{T_1} \approx q_{T_1}$, an iterative application of Proposition 4 implies that there is a safe run from q_{T_1} to some state q such that $q \approx s_{T_2}$. Since the run from q_{T_1} to q is safe, the states q_{T_1} and q are in the same safe component, and so $q \in T_1$. Since $q_{T_2} \approx s_{T_2}$, then $q \approx q_{T_2}$. Since $\mathcal{A}$ is safe-centralized, the latter implies that q and q_{T_2} are in the same safe component, and so $q \in T_2$, and we have reached a contradiction.

It is left to prove that for every safe component $T \in S(\mathcal{A})$, it holds that $|T| \le |\eta(T)|$. Let $T \in S(\mathcal{A})$ be a safe component of $\mathcal{A}$. By the definition of η, there are $q_T \in T$ and $s_T \in \eta(T)$ such that $q_T \approx s_T$. Since $\mathcal{A}$ is normal, there is a safe run $q_0, q_1, \ldots q_m$ of $\mathcal{A}$ that starts in q_T and traverses all the states in T. Since $\mathcal{A}$ is safe-minimal, no two states in T are strongly equivalent. Therefore, there is a subset $I \subseteq \{0, 1, \ldots, m\}$ of indices, with $|I| = |T|$, such that for every two different indices $i_1, i_2 \in I$, it holds that $q_{i_1} \not\approx q_{i_2}$. By applying Proposition 4 iteratively, there is a safe run $s_0, s_1, \ldots s_m$ of $\mathcal{B}$ that starts in s_T and such that for every $0 \le i \le m$, it holds that $q_i \approx s_i$. Since the run is safe, it stays in $\eta(T)$. Then, however, for every two different indices $i_1, i_2 \in I$, we have that $s_{i_1} \not\approx s_{i_2}$, and so $s_{i_1} \neq s_{i_2}$. Hence, $|\eta(T)| \ge |I| = |T|$. ◀

We can now prove that the additional two properties imply the minimality of nice GFG-tNCWs.

▶ **Theorem 8.** *Consider a nice GFG-tNCW $\mathcal{A}$. If $\mathcal{A}$ is safe-centralized and safe-minimal, then $\mathcal{A}$ is a minimal GFG-tNCW for $L(\mathcal{A})$.*

Proof. Let $\mathcal{B}$ be a GFG-tNCW equivalent to $\mathcal{A}$. By Theorem 2, we can assume that $\mathcal{B}$ is nice. Indeed, otherwise we can make it nice without increasing its state space. Then, by Lemma 7, there is an injection $\eta : S(\mathcal{A}) \to S(\mathcal{B})$ such that for every safe component $T \in S(\mathcal{A})$, it holds that $|T| \le |\eta(T)|$. Hence,

$$|\mathcal{A}| = \sum_{T \in S(\mathcal{A})} |T| \le \sum_{T \in S(\mathcal{A})} |\eta(T)| \le \sum_{T' \in S(\mathcal{B})} |T'| = |\mathcal{B}|.$$

Indeed, the first inequality follows from the fact $|T| \le |\eta(T)|$, and the second inequality follows from the fact that η is injective. ◀

▶ Remark 9. Recall that we assume that the transition function of GFG-tNCWs is total. Clearly, a non-total GFG-tNCW can be made total by adding a rejecting sink. One may wonder whether the additional state that this process involves interferes with our minimality proof. The answer is negative: if $\mathcal{B}$ in Theorem 8 is not total, then, by Proposition 5, $\mathcal{A}$ has a state s such that $q_{rej} \precsim s$, where q_{rej} is a rejecting sink we need to add to $\mathcal{B}$ if we want to make it total. Thus, $L(\mathcal{A}^s) = \emptyset$, and we may not count it if we allow GFG-tNCWs without a total transition function.

4 Safe Centralization

Consider a nice GFG-tNCW $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, q^0_\mathcal{A}, \delta_\mathcal{A}, \alpha_\mathcal{A} \rangle$. Recall that $\mathcal{A}$ is safe-centralized if for every two states $q, s \in Q_\mathcal{A}$, if $q \precsim s$, then q and s are in the same safe component. In this section we describe how to turn a given nice GFG-tNCW into a nice safe-centralized GFG-tNCW. The resulted tNCW is also going to be *α-homogenous*: for every state $q \in Q_\mathcal{A}$ and letter $\sigma \in \Sigma$, either $\delta^\alpha_\mathcal{A}(q, \sigma) = \emptyset$ or $\delta^{\bar\alpha}_\mathcal{A}(q, \sigma) = \emptyset$.

Let $H \subseteq S(\mathcal{A}) \times S(\mathcal{A})$ be such that for all safe components $S, S' \in S(\mathcal{A})$, we have that $H(S, S')$ iff there exist states $q \in S$ and $q' \in S'$ such that $q \precsim q'$. That is, when $S \neq S'$, then the states q and q' witness that $\mathcal{A}$ is not safe-centralized. Recall that $q \precsim q'$ iff $L(\mathcal{A}^q) = L(\mathcal{A}^{q'})$ and $L_{safe}(\mathcal{A}^q) \subseteq L_{safe}(\mathcal{A}^{q'})$. Since language containment for GFG-tNCWs can be checked in polynomial time [12, 17], the first condition can be checked in polynomial time. Since $\mathcal{A}$ is safe deterministic, the second condition reduces to language containment between deterministic automata and can also be checked in polynomial time. Hence, the relation H can be computed in polynomial time.

▶ **Lemma 10.** *Consider safe components $S, S' \in S(\mathcal{A})$ such that $H(S, S')$. Then, for every $p \in S$ there is $p' \in S'$ such that $p \precsim p'$.*

Proof. Since $H(S, S')$, then, by definition, there are states $q \in S$ and $q' \in S'$ such that $q \precsim q'$. Let p be a state in S. Since $\mathcal{A}$ is normal, there is a safe run from q to p in S. Since $q \precsim q'$, an iterative application of Proposition 4 implies that there is a safe run from q' to some state p' in S' for which $p \precsim p'$, and we are done. ◀

▶ **Lemma 11.** *The relation H is transitive: for every safe components $S, S', S'' \in S(\mathcal{A})$, if $H(S, S')$ and $H(S', S'')$, then $H(S, S'')$.*

Proof. Let $S, S', S'' \in S(\mathcal{A})$ be safe components of $\mathcal{A}$ such that $H(S, S')$ and $H(S', S'')$. Since, $H(S, S')$, there are states $q \in S$ and $q' \in S'$ such that $q \precsim q'$. Now, since $H(S', S'')$, we get by Lemma 10 that that for all states in S', in particular for q', there is a state $q'' \in S''$ such that $q' \precsim q''$. The transitivity of $\precsim$ then implies that $q \precsim q''$, and so $H(S, S'')$. ◀

We say that a set $\mathcal{S} \subseteq S(\mathcal{A})$ is a *frontier of $\mathcal{A}$* if for every safe component $S \in S(\mathcal{A})$, there is a safe component $S' \in \mathcal{S}$ with $H(S, S')$, and for all safe components $S, S' \in \mathcal{S}$ such that $S \neq S'$, we have that $\neg H(S, S')$ and $\neg H(S', S)$. Once H is calculated, a frontier of $\mathcal{A}$ can be found in linear time. For example, as H is transitive, we can take one vertex from each ergodic SCC in the graph $\langle S(\mathcal{A}), H\rangle$. Note that all frontiers of $\mathcal{A}$ are of the same size, namely the number of ergodic SCCs in this graph.

Given a frontier $\mathcal{S}$ of $\mathcal{A}$, we define the automaton $\mathcal{B}_{\mathcal{S}} = \langle \Sigma, Q_{\mathcal{S}}, q^0_{\mathcal{S}}, \delta_{\mathcal{S}}, \alpha_{\mathcal{S}}\rangle$, where $Q_{\mathcal{S}} = \{q \in Q_{\mathcal{A}} : q \in S \text{ for some } S \in \mathcal{S}\}$, and the other components are defined as follows. The initial state $q^0_{\mathcal{S}}$ is chosen such that $q^0_{\mathcal{S}} \sim_{\mathcal{A}} q^0_{\mathcal{A}}$. Specifically, if $q^0_{\mathcal{A}} \in Q_{\mathcal{S}}$, we take $q^0_{\mathcal{S}} = q^0_{\mathcal{A}}$. Otherwise, by Lemma 10 and the definition of $\mathcal{S}$, there is a state $q' \in Q_{\mathcal{S}}$ such that $q^0_{\mathcal{A}} \precsim q'$, and we take $q^0_{\mathcal{S}} = q'$. The transitions in $\mathcal{B}_{\mathcal{S}}$ are either $\bar{\alpha}$-transitions of $\mathcal{A}$, or α-transitions that we add among the safe components in $\mathcal{S}$ in a way that preserves language equivalence. Formally, consider a state $q \in Q_{\mathcal{S}}$ and a letter $\sigma \in \Sigma$. If $\delta^{\bar{\alpha}}_{\mathcal{A}}(q, \sigma) \neq \emptyset$, then $\delta^{\bar{\alpha}}_{\mathcal{S}}(q, \sigma) = \delta^{\bar{\alpha}}_{\mathcal{A}}(q, \sigma)$ and $\delta^{\alpha}_{\mathcal{S}}(q, \sigma) = \emptyset$. If $\delta^{\bar{\alpha}}_{\mathcal{A}}(q, \sigma) = \emptyset$, then $\delta^{\bar{\alpha}}_{\mathcal{S}}(q, \sigma) = \emptyset$ and $\delta^{\alpha}_{\mathcal{S}}(q, \sigma) = \{q' \in Q_{\mathcal{S}} : \text{ there is } q'' \in \delta^{\alpha}_{\mathcal{A}}(q, \sigma) \text{ such that } q' \sim_{\mathcal{A}} q''\}$. Note that $\mathcal{B}_{\mathcal{S}}$ is α-homogenous.

▶ **Example 12.** Consider the tDCW $\mathcal{A}$ appearing in Figure 1. Recall that the dashed transitions are α-transitions. Since $\mathcal{A}$ is normal and deterministic, it is nice. By removing the α-transitions of $\mathcal{A}$, we get the safe components described in in Figure 2. Since $q_2 \precsim q_0$, we have that $\mathcal{A}$ has a single frontier $\mathcal{S} = \{\{q_0, q_1\}\}$. The automaton $\mathcal{B}_{\mathcal{S}}$ appears in Figure 3. As all the states of $\mathcal{A}$ are equivalent, we direct a σ-labeled α-transition to q_0 and to q_1, for every state with no σ-labeled transition in $\mathcal{S}$.

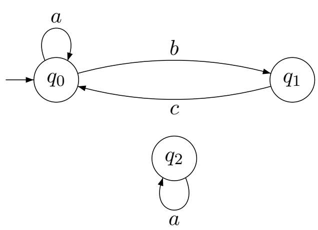

Figure 2 The safe components of $\mathcal{A}$.

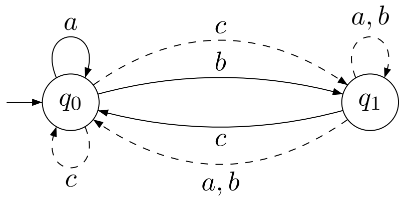

Figure 3 The tNCW $\mathcal{B}_{\{\{q_0,q_1\}\}}$.

We extend Proposition 1 to the setting of $\mathcal{A}$ and $\mathcal{B}_{\mathcal{S}}$:

▶ **Proposition 13.** *Consider states q and s of $\mathcal{A}$ and $\mathcal{B}_{\mathcal{S}}$, respectively, a letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, q'\rangle$ and $\langle s, \sigma, s'\rangle$ of $\mathcal{A}$ and $\mathcal{B}_{\mathcal{S}}$, respectively. If $q \sim_{\mathcal{A}} s$, then $q' \sim_{\mathcal{A}} s'$.*

Proof. If $\langle s, \sigma, s'\rangle$ is an $\bar{\alpha}$-transition of $\mathcal{B}_{\mathcal{S}}$, then, by the definition of $\Delta_{\mathcal{S}}$, it is also an $\bar{\alpha}$-transition of $\mathcal{A}$. Hence, since $q \sim_{\mathcal{A}} s$ and $\mathcal{A}$ is nice, in particular, semantically deterministic, we get by Proposition 1 that $q' \sim_{\mathcal{A}} s'$. If $\langle s, \sigma, s'\rangle$ is an α-transition of $\mathcal{B}_{\mathcal{S}}$, then, by the definition of $\Delta_{\mathcal{S}}$, there is some

$s'' \in \delta_{\mathcal{A}}(s,\sigma)$ with $s' \sim_{\mathcal{A}} s''$. Again, since $q \sim_{\mathcal{A}} s$ and $\mathcal{A}$ is semantically deterministic, we have by Proposition 1 that $s'' \sim_{\mathcal{A}} q'$, and thus $s' \sim_{\mathcal{A}} q'$. ◀

▶ **Proposition 14.** *Let q and s be states of $\mathcal{A}$ and $\mathcal{B}_{\mathcal{S}}$, respectively, with $q \sim_{\mathcal{A}} s$. It holds that $\mathcal{B}^s_{\mathcal{S}}$ is a GFG-tNCW equivalent to $\mathcal{A}^q$.*

Proof. We first prove that $L(\mathcal{B}^s_{\mathcal{S}}) \subseteq L(\mathcal{A}^q)$. Consider a word $w = \sigma_1\sigma_2\ldots \in L(\mathcal{B}^s_{\mathcal{S}})$. Let $s_0, s_1, s_2, \ldots$ be an accepting run of $\mathcal{B}^s_{\mathcal{S}}$ on w. Then, there is $i \geq 0$ such that $s_i, s_{i+1}, \ldots$ is a safe run of $\mathcal{B}^{s_i}_{\mathcal{S}}$ on the suffix $w[i+1,\infty]$. Let $q_0, q_1, \ldots q_i$ be a run of $\mathcal{A}^q$ on the prefix $w[1,i]$. Since $q_0 \sim_{\mathcal{A}} s_0$, we get, by an iterative application of Proposition 13, that $q_i \sim_{\mathcal{A}} s_i$. In addition, as the run of $\mathcal{B}^{s_i}_{\mathcal{S}}$ on the suffix $w[i+1,\infty]$ is safe, it is also a safe run of $\mathcal{A}^{s_i}$. Hence, $w[i+1,\infty] \in L(\mathcal{A}^{q_i})$, and thus $q_0, q_1, \ldots, q_i$ can be extended to an accepting run of $\mathcal{A}^q$ on w.

Next, we prove that $L(\mathcal{A}^q) \subseteq L(\mathcal{B}^s_{\mathcal{S}})$ and that $\mathcal{B}^s_{\mathcal{S}}$ is a GFG-tNCW. We do this by defining a strategy $g : \Sigma^* \to Q_{\mathcal{S}}$ such that for all words $w \in L(\mathcal{A}^q)$, we have that $g(w)$ is an accepting run of $\mathcal{B}^s_{\mathcal{S}}$ on w. First, $g(\epsilon) = s$. Then, for $u \in \Sigma^*$ and $\sigma \in \Sigma$, we define $g(u\cdot\sigma)$ as follows. Recall that $\mathcal{A}$ is nice. So, in particular, $\mathcal{A}^q$ is GFG. Let f be a strategy witnessing $\mathcal{A}^q$'s GFGness. If $\delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma) \neq \emptyset$, then $g(u\cdot\sigma) = q'$ for some $q' \in \delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma)$. If $\delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma) = \emptyset$, then $g(u\cdot\sigma) = q'$ for some state $q' \in Q_{\mathcal{S}}$ such that $f(u\cdot\sigma) \precsim_{\mathcal{A}} q'$. Note that since $\mathcal{S}$ is a frontier, such a state q' exists. We prove that g is consistent with $\Delta_{\mathcal{S}}$. In fact, we prove a stronger claim, namely for all $u \in \Sigma^*$ and $\sigma \in \Sigma$, we have that $f(u) \sim_{\mathcal{A}} g(u)$ and $\langle g(u), \sigma, g(u\cdot\sigma)\rangle \in \Delta_{\mathcal{S}}$.

The proof proceeds by an induction on $|u|$. For this induction base, as $f(\epsilon) = q$, $g(\epsilon) = s$, and $q \sim_{\mathcal{A}} s$, we are done. Given u and σ, consider a transition $\langle g(u), \sigma, s'\rangle \in \Delta_{\mathcal{S}}$. Since $\mathcal{B}_{\mathcal{S}}$ is total, such a transition exists. We distinguish between two cases. If $\delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma) \neq \emptyset$, then, as $\mathcal{B}_{\mathcal{S}}$ is α-homogenous and safe deterministic, the state s' is the only state in $\delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma)$. Hence, by the definition of g, we have that $g(u\cdot\sigma) = s'$ and so $\langle g(u), \sigma, g(u\cdot\sigma)\rangle \in \Delta_{\mathcal{S}}$. If $\delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma) = \emptyset$, we claim that $g(u\cdot\sigma) \sim_{\mathcal{A}} s'$ Then, as $s' \in \delta^{\alpha}_{\mathcal{S}}(g(u),\sigma)$, the definition of $\Delta_{\mathcal{S}}$ for the case $\delta^{\bar{\alpha}}_{\mathcal{S}}(g(u),\sigma) = \emptyset$ implies that $\langle g(u), \sigma, g(u\cdot\sigma)\rangle \in \Delta_{\mathcal{S}}$. By the induction hypothesis, we have that $f(u) \sim_{\mathcal{A}} g(u)$. Hence, as $\langle f(u), \sigma, f(u\cdot\sigma))\rangle \in \delta_{\mathcal{A}}$ and $\langle g(u), \sigma, s'\rangle \in \Delta_{\mathcal{S}}$, we have, by Proposition 13, that $f(u\cdot\sigma) \sim_{\mathcal{A}} s'$. Recall that g is defined so that $f(u\cdot\sigma) \precsim_{\mathcal{A}} g(u\cdot\sigma)$. In particular, $f(u\cdot\sigma) \sim_{\mathcal{A}} g(u\cdot\sigma)$. Hence, by transitivity of $\sim_{\mathcal{A}}$, we have that $g(u\cdot\sigma) \sim_{\mathcal{A}} s'$. In addition, by the induction hypothesis, we have that $f(u) \sim_{\mathcal{A}} g(u)$, and so, in both cases, Proposition 13 implies that $f(u\cdot\sigma) \sim_{\mathcal{A}} g(u\cdot\sigma)$.

It is left to prove that for every infinite word $w = \sigma_1\sigma_2\ldots \in \Sigma^\omega$, if $w \in L(\mathcal{A}^q)$, then $g(w)$ is accepting. Assume that $w \in L(\mathcal{A}^q)$ and consider the run $f(w)$ of $\mathcal{A}^q$ on w. Since $f(w)$ is accepting, there is $i \geq 0$ such that $f(w[1,i]), f(w[1,i+1])\ldots$ is a safe run of $\mathcal{A}^{f(w[1,i])}$ on the suffix $w[i+1,\infty]$. We prove that $g(w)$ may traverse at most one α-transition when it reads the suffix $w[i+1,\infty]$. Assume that there is some $j \geq i$ such that $\langle g(w[1,j]), \sigma_{j+1}, g(w[1,j+1])\rangle \in \alpha_{\mathcal{S}}$. Then, by g's definition, we have that $f(w[1,j+1]) \precsim_{\mathcal{A}} g(w[1,j+1])$. Therefore, as $\mathcal{B}_{\mathcal{S}}$ follows the safe components in $\mathcal{S}$, we have that $L_{safe}(\mathcal{A}^{f(w[1,j+1])}) \subseteq L_{safe}(\mathcal{A}^{g(w[1,j+1])}) = L_{safe}(\mathcal{B}^{g(w[1,j+1])}_{\mathcal{S}})$, and thus $w[j+2,\infty] \in L_{safe}(\mathcal{B}^{g(w[1,j+1])}_{\mathcal{S}})$. Since $\mathcal{B}_{\mathcal{S}}$ is α-homogenous and safe-deterministic, there is a single run of $\mathcal{B}^{g(w[1,j+1])}_{\mathcal{S}}$ on $w[j+2,\infty]$, and this is the run that g follows. Therefore, $g(w[1,j+1]), g(w[1,j+2]), \ldots$ is a safe run, and we are done. ◀

▶ **Proposition 15.** *For every frontier $\mathcal{S}$, the GFG-tNCW $\mathcal{B}_{\mathcal{S}}$ is nice, safe-centralized, and α-homogenous.*

Proof. It is easy to see that the fact $\mathcal{A}$ is nice implies that $\mathcal{B}_{\mathcal{S}}$ is normal and safe deterministic. It can be shown that all the states in $\mathcal{B}_{\mathcal{S}}$ are reachable, yet anyway states that are nonreachable are easy to detect and their removal affects neither $\mathcal{B}_{\mathcal{S}}$'s language nor its other properties. Finally, Proposition 14 implies that all its states are GFG. To conclude that $\mathcal{B}_{\mathcal{S}}$ is nice, we prove below that it is semantically deterministic. Consider transitions $\langle q, \sigma, s_1\rangle$ and $\langle q, \sigma, s_2\rangle$ in $\Delta_{\mathcal{S}}$. We need to show that $s_1 \sim_{\mathcal{B}_{\mathcal{S}}} s_2$.

By the definition of $\Delta_{\mathcal{S}}$, there are transitions $\langle q, \sigma, s'_1\rangle$ and $\langle q, \sigma, s'_2\rangle$ in $\Delta_{\mathcal{A}}$ for states s'_1 and s'_2 such that $s_1 \sim_{\mathcal{A}} s'_1$ and $s_2 \sim_{\mathcal{A}} s'_2$. As $\mathcal{A}$ is semantically deterministic, we have that $s'_1 \sim_{\mathcal{A}} s'_2$, thus by transitivity of $\sim_{\mathcal{A}}$, we get that $s_1 \sim_{\mathcal{A}} s_2$. Then, Proposition 14 implies that $L(\mathcal{A}^{s_1}) = L(\mathcal{B}_{\mathcal{S}}^{s_1})$ and $L(\mathcal{A}^{s_2}) = L(\mathcal{B}_{\mathcal{S}}^{s_2})$, and so we get that $s_1 \sim_{\mathcal{B}_{\mathcal{S}}} s_2$. Thus, $\mathcal{B}_{\mathcal{S}}$ is semantically deterministic.

As we noted in the definition of its transitions, $\mathcal{B}_{\mathcal{S}}$ is α-homogenous. It is thus left to prove that $\mathcal{B}_{\mathcal{S}}$ is safe-centralized. Let q and s be states of $\mathcal{B}_{\mathcal{S}}$ such that $q \precsim_{\mathcal{B}_{\mathcal{S}}} s$; that is, $L(\mathcal{B}_{\mathcal{S}}^q) = L(\mathcal{B}_{\mathcal{S}}^s)$ and $L_{safe}(\mathcal{B}_{\mathcal{S}}^q) \subseteq L_{safe}(\mathcal{B}_{\mathcal{S}}^s)$. Let $S, T \in \mathcal{S}$ be the safe components of q and s, respectively. We need to show that $S = T$. By Proposition 14, we have that $L(\mathcal{A}^q) = L(\mathcal{B}_{\mathcal{S}}^q)$ and $L(\mathcal{A}^s) = L(\mathcal{B}_{\mathcal{S}}^s)$. As $\mathcal{B}_{\mathcal{S}}$ follows the safe components in $\mathcal{S}$, we have that $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{B}_{\mathcal{S}}^q)$ and $L_{safe}(\mathcal{A}^s) = L_{safe}(\mathcal{B}_{\mathcal{S}}^s)$. Hence, $q \precsim_{\mathcal{A}} s$, implying $H(S, T)$. Since $\mathcal{S}$ is a frontier, this is possible only when $S = T$. ◀

▶ **Theorem 16.** *Every nice GFG-tNCW can be turned in polynomial time into an equivalent nice, safe-centralized, and α-homogenous GFG-tNCW.*

5 Safe Minimization

In the setting of finite words, a *quotient automaton* is obtained by merging equivalent states, and is guaranteed to be minimal. In the setting of co-Büchi automata, it may not be possible to define an equivalent language on top of the quotient automaton. For example, all the states in the GFG-tNCW $\mathcal{A}$ in Figure 1 are equivalent, and still it is impossible to define its language on top of a single-state tNCW. In this section we show that when we start with a nice, safe-centralized, and α-homogenous GFG-tNCW $\mathcal{B}$, the transition to a quotient automaton, namely merging of strongly-equivalent states, is well defined and results in a GFG-tNCW equivalent to $\mathcal{B}$ that attains all the helpful properties of $\mathcal{B}$, and is also safe minimal[4]. By Theorem 8, it is also minimal.

Consider a nice, safe-centralized, and α-homogenous GFG-tNCW $\mathcal{B} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$. For a state $q \in Q$, define $[q] = \{q' \in Q : q \approx_{\mathcal{B}} q'\}$. We define the tNCW $\mathcal{C} = \langle \Sigma, Q_{\mathcal{C}}, [q_0], \delta_{\mathcal{C}}, \alpha_{\mathcal{C}}\rangle$, where $Q_{\mathcal{C}} = \{[q] : q \in Q\}$, the transition function is such that $\langle [q], \sigma, [p]\rangle \in \Delta_{\mathcal{C}}$ iff there are $q' \in [q]$ and $p' \in [p]$ such that $\langle q', \sigma, p'\rangle \in \Delta$, and $\langle [q], \sigma, [p]\rangle \in \alpha_{\mathcal{C}}$ iff $\langle q', \sigma, p'\rangle \in \alpha$. Note that $\mathcal{B}$ being α-homogenous implies that $\alpha_{\mathcal{C}}$ is well defined; that is, independent of the choice of q' and p'. To see why, assume that $\langle q', \sigma, p'\rangle \in \bar{\alpha}$ and let q'' be a state in $[q]$. As $q' \approx_{\mathcal{B}} q''$, we have by Proposition 4 that there is $p'' \in [p]$ such that $\langle q'', \sigma, p''\rangle \in \bar{\alpha}$. Thus, as $\mathcal{B}$ is α-homogenous, there is no σ-labeled α-transition from q'' to a state in $[p]$. Note that we have proved that if $\langle [q], \sigma, [p]\rangle$ is an $\bar{\alpha}$-transition of $\mathcal{C}$, then for every $q' \in [q]$, there is $p' \in [p]$ such that $\langle q', \sigma, p'\rangle$ is an $\bar{\alpha}$-transition of $\mathcal{B}$, and thus the $\supseteq$-direction of the following observation, suggesting that a safe run in $\mathcal{C}$ induces a safe run in $\mathcal{B}$, follows by a simple induction. The $\subseteq$-direction follows immediately from the definition of $\mathcal{C}$.

▶ **Observation 17.** *For every $[p] \in Q_{\mathcal{C}}$ and every $s \in [p]$, it holds that $L_{safe}(\mathcal{B}^s) = L_{safe}(\mathcal{C}^{[p]})$.*

We extend Propositions 1 and 13 to the setting of $\mathcal{B}$ and $\mathcal{C}$:

▶ **Proposition 18.** *Consider states $s \in Q$ and $[p] \in Q_{\mathcal{C}}$, a letter $\sigma \in \Sigma$, and transitions $\langle s, \sigma, s'\rangle$ and $\langle [p], \sigma, [p']\rangle$ of $\mathcal{B}$ and $\mathcal{C}$, respectively. If $s \sim p$, then $s' \sim p'$.*

Proof. As $\langle [p], \sigma, [p']\rangle$ is a transition of $\mathcal{C}$, there are states $t \in [p]$ and $t' \in [p']$, such that $\langle t, \sigma, t'\rangle \in \Delta$. If $s \sim p$, then $s \sim t$. Since $\mathcal{B}$ is nice, in particular, semantically deterministic, and $\langle s, \sigma, s'\rangle \in \Delta$, we get by Proposition 1 that $s' \sim t'$. Thus, as $t' \sim p'$, we are done. ◀

[4] In fact, α-homogeneity is not required, but as the GFG-tNCW $\mathcal{B}_{\mathcal{S}}$ obtained in Section 4 is α-homogenous, which simplifies the proof, we are going to rely on it.

▶ **Proposition 19.** *For every $[p] \in Q_{\mathcal{C}}$ and $s \in [p]$, we have that $\mathcal{C}^{[p]}$ is a GFG-tNCW equivalent to $\mathcal{B}^s$.*

Proof. We first prove that $L(\mathcal{C}^{[p]}) \subseteq L(\mathcal{B}^s)$. Consider a word $w = \sigma_1\sigma_2 \ldots \in L(\mathcal{C}^{[p]})$. Let $[p_0], [p_1], [p_2], \ldots$ be an accepting run of $\mathcal{C}^{[p]}$ on w. Then, there is $i \geq 0$ such that $[p_i], [p_{i+1}], \ldots$ is a safe run of $\mathcal{C}^{[p_i]}$ on the suffix $w[i+1, \infty]$. Let $s_0, s_1, \ldots s_i$ be a run of $\mathcal{B}^s$ on the prefix $w[1, i]$. Note that $s_0 = s$. Since $s_0 \in [p_0]$, we have that $s_0 \sim p_0$, and thus an iterative application of Proposition 18 implies that $s_i \sim p_i$. In addition, as $w[i+1, \infty]$ is in $L_{safe}(\mathcal{C}^{[p_i]})$, we get, by Observation 17, that $w[i+1, \infty] \in L_{safe}(\mathcal{B}^{p_i})$. Since $L_{safe}(\mathcal{B}^{p_i}) \subseteq L(\mathcal{B}^{p_i})$ and $s_i \sim p_i$, we have that $w[i+1, \infty] \in L(\mathcal{B}^{s_i})$. Hence, $s_0, s_1, \ldots s_i$ can be extended to an accepting run of $\mathcal{B}^s$ on w.

Next, we prove that $L(\mathcal{B}^s) \subseteq L(\mathcal{C}^{[p]})$ and that $\mathcal{C}^{[p]}$ is a GFG-tNCW. We do this by defining a strategy $h : \Sigma^* \to Q_{\mathcal{C}}$ such that for all words $w \in L(\mathcal{B}^s)$, we have that $h(w)$ is an accepting run of $\mathcal{C}^{[p]}$ on w. We define h as follows. Recall that $\mathcal{B}$ is nice. So, in particular, $\mathcal{B}^s$ is GFG. Let g be a strategy witnessing $\mathcal{B}^s$'s GFGness. We define $h(u) = [g(u)]$, for every finite word $u \in \Sigma^*$. Consider a word $w \in L(\mathcal{B}^s)$, and consider the accepting run $g(w) = g(w[1, 0]), g(w[1, 1]), g(w[1, 2]), \ldots$ of $\mathcal{B}^s$ on w. Note that by the definition of $\mathcal{C}$, we have that $h(w) = [g(w[1, 0])], [g(w[1, 1])], [g(w[1, 2])], \ldots$ is an accepting run of $\mathcal{C}^{[p]}$ on w, and so we are done. ◀

▶ **Proposition 20.** *The GFG-tNCW $\mathcal{C}$ is nice, safe-centralized, and safe-minimal.*

The proof of the proposition is in the full version. The considerations are similar to those in the proof of Proposition 15. In particular, for safe minimality, note that for states q and s of $\mathcal{B}$, we have that $[q] \approx [s]$ iff $[q] \precsim [s]$ and $[s] \precsim [q]$. Thus, it is sufficient to prove that if $[q] \precsim [s]$ then $q \precsim s$. Thus, we can now conclude the following:

▶ **Theorem 21.** *Every nice, safe-centralized, and α-homogenous GFG-tNCW can be turned in polynomial time into an equivalent nice, safe-centralized, and safe-minimal GFG-tNCW.*

6 Discussion

We presented a polynomial minimization algorithm for GFG-tNCWs. In contrast, minimization of DCWs is NP-complete [33]. This raises a natural question, as to whether both relaxations of the problem, namely the consideration of GFG automata, rather than deterministic ones, and the consideration of transition-based acceptance, rather than state-based one, are crucial for efficiency. Our conjecture is that minimization of transition-based DCWs (and hence, also transition-based DBWs) can be solved in polynomial time. Thus, the relaxation to GFG is not needed. Our conjecture is based on the understanding that the quotient construction fails for automata on infinite words as it does not capture traversal of transitions. Moreover, the study of GFG automata so far shows that their behavior is similar to that of deterministic automata. In particular, it is not hard to see that the NP-hardness proof of Schewe for DBWs minimization applies also to GFG-NBWs. The use of transition-based acceptance is related to another open problem in the context of DBW minimization: is there a 2-approximation polynomial algorithm for it, that is one that generates a DBW that is at most twice as big as a minimal one. Note that a tight minimization for the transition-based case would imply a positive answer here. Note also that the vertex-cover problem, used in Schewe's reduction has a polynomial 2-approximation. As described in Section 1, there is recently growing use of automata with transition-based acceptance. Our work here is another evidence to their usefulness.

We find the study of minimization of GFG automata of interest also beyond being an intermediate result in the quest for efficient transition-based DBW minimization. Indeed, GFG automata are important in practice, as they are used in synthesis and control, and in the case of the co-Büchi acceptance condition, they may be exponentially more succinct than their deterministic equivalences.

Another open problem, which is interesting from both the theoretical and practical points of view, is minimization of GFG-tNBW. Note that unlike the deterministic case, GFG-tNBW and GFG-tNCW are not dual. Also, experience shows that algorithms for GFG-tNBW and GFG-tNCW are quite different [2–4, 17].

Finally, recall that there may be different minimal tDCWs for a given language of infinite words. Our results show that the picture for minimal GFG-tNCWs is cleaner: Consider a language $L \subseteq \Sigma^\omega$, and let $\mathcal{A}$ be a minimal GFG-tNCW for L obtained by safe-centralizing and safe-minimizing a nice GFG-tNCW for it. Consider a nice minimal GFG-tNCW $\mathcal{B}$ for L. Then, the injection $\eta : S(\mathcal{A}) \to S(\mathcal{B})$ from Lemma 7 is actually a *bijection*; that is, η is one-to-one and onto. Indeed, for every safe component $T \in S(\mathcal{A})$ it holds that $|T| = |\eta(T)|$. Moreover, as both $\mathcal{A}$ and $\mathcal{B}$ are nice, related safe components are *isomorphic*, thus there is an bijection $\kappa : Q_\mathcal{A} \to Q_\mathcal{B}$ such that for every $q \in Q_\mathcal{A}$, we have that $q \approx \kappa(q)$, and for every $\bar{\alpha}$-transition $\langle q, \sigma, s\rangle$ of $\mathcal{A}$, we have that $\langle \kappa(q), \sigma, \kappa(s)\rangle$ is an $\bar{\alpha}$-transition of $\mathcal{B}$. Thus, all nice minimal GFG-tNCWs for L have the same set of safe components, and they differ only in α-transitions among these safe components. An interesting research direction is a study of these safe components and in particular a characterization of L by a congruence-based relation on finite words that is induced by them.

References

1. A. Badr, V. Geffert, and I. Shipman. Hyper-minimizing minimized deterministic finite state automata. *ITA*, 43(1):69–94, 2009.
2. M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 122 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
3. 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.
4. U. Boker, O. Kupferman, and M. Skrzypczak. How deterministic are Good-For-Games automata? In *Proc. 37th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 93 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, 2017.
5. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
6. Th. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.
7. A. Duret-Lutz, A. Lewkowicz, A. Fauchille, Th. Michaud, E. Renault, and L. Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *14th Int. Symp. on Automated Technology for Verification and Analysis*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, 2016.
8. J. Esparza, O. Kupferman, and M.Y. Vardi. Verification. In *Handbook AutoMathA*, pages 549–588. European Mathematical Society, 2018.
9. D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Proc. 22nd International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2002.
10. S. Gurumurthy, R. Bloem, and F. Somenzi. Fair simulation minimization. In *Proc. 14th Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 610–623. Springer, 2002.
11. S. Halamish and O. Kupferman. Minimizing deterministic lattice automata. *ACM Transactions on Computational Logic*, 16(1):1–21, 2015.
12. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.

13 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.

14 J.E. Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

15 A. Jez and A. Maletti. Hyper-minimization for deterministic tree automata. *Int. J. Found. Comput. Sci.*, 24(6):815–830, 2013.

16 T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

17 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.

18 O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 199 – 213. Springer, 2007.

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

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

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

22 W. Li, Sh. Kan, and Z. Huang. A better translation from LTL to transition-based generalized büchi automata. *IEEE Access*, 5:27081–27090, 2017.

23 C. Löding. Efficient minimization of deterministic weak omega-automata. *Information Processing Letters*, 79(3):105–109, 2001.

24 A. Malcher. Minimizing finite automata is computationally hard. *Theoretical Computer Science*, 327(3):375–390, 2004.

25 M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

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

27 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.

28 J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.

29 A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

30 D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th Symp. on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*. Springer, 1998.

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

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

33 S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.

34 S. Sickert, J. Esparza, S. Jaax, and J. Křetínský. Limit-deterministic büchi automata for linear temporal logic. In *Proc. 28th Int. Conf. on Computer Aided Verification*, volume 9780 of *Lecture Notes in Computer Science*, pages 312–332. Springer, 2016.

35 R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.

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