Minimization of Automata for Liveness Languages

Bader Abu Radi and Orna Kupferman

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel,
bader.aburadi@gmail.com, orna@cs.huji.ac.il

Abstract. While the *minimization problem* for deterministic Büchi word automata is known to be NP-complete, several fundamental problems around it are still open. This includes the complexity of minimzation for *transition-based* automata, where acceptance is defined with respect to the set of transitions that a run traverses infinitely often, and minimization for *good-for-games* (GFG) automata, where nondeterminism is allowed, yet has to be resolved in a way that only depends on the past.
Of special interest in formal verification are *liveness* properties, which state that something "good" eventually happens. Liveness languages constitute a strict fragment of ω-regular languages, which suggests that minimization of automata recognizing liveness languages may be easier, as is the case for languages recognizable by weak automata, in particular safety languages. We define three classes of liveness, and study the minimization problem for automata recognizing languages in the classes. Our results refer to the basic minimization problem as well as to its extension to transition-based and GFG automata. In some cases, we provide bounds, and in others we provide connections between the different settings. Thus, our results are of practical interest and also improve our understanding of the (still very mysterious) minimization problem.

Keywords: Automata on infinite words, Minimization, Complexity, Good-for-games automata, Büchi, Liveness

1 Introduction

A prime application of *automata theory* is specification, verification, and synthesis of reactive systems [36,12]. 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 [9] (NBWs and DBWs, 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 (NCWs and DCWs), a run is accepting iff it visits α only finitely often.

A classical problem in automata theory is *minimization*: the generation of an equivalent automaton with a minimal number of states. For NBWs and NCWs, minimization is PSPACE-complete, as it is for nondeterministic automata on finite words [20]. For DBWs and DCWs, minimization is NP-complete [33]. Thus, the transition to infinite words makes the problem harder. Indeed, for deterministic automata on finite words (DFWs), a minimization algorithm, based on the Myhill-Nerode right congruence [29,30], generates in polynomial time a canonical minimal deterministic automaton [18].

While [33] solves the question of the complexity of DBW and DCW minimization, several fundamental questions around the problem are still open. This includes the complexity of minimzation for *transition-based* automata and *good-for-games* (GFG) automata.

In transition-based automata, acceptance is defined with respect to the set of transitions that a run traverses infinitely often. In particular, in Büchi automata (tNBWs and tDBWs), the acceptance condition is a subset α of transitions, and a run is required to traverse transitions in α infinitely often, and dually for co-Büchi. 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 [14,15,11,35,26].

In GFG automata, nondeterminism is allowed, yet has to be resolved in a way that only depends on the past. Consequently, GFG automata can be used instead of deterministic automata in games whose winning condition is specified by an ω-regular language [23,17,10]. Such games are extensively used in the solution of the *synthesis* problem from LTL or ω-regular specifications [32,4]. 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. Surprisingly, however, GFG-NBWs need not be DBP [5]. Moreover, the best known determinization construction for GFG-NBWs is quadratic, whereas determinization of GFG-NCWs has a tight exponential bound [21]. Thus, GFG automata on infinite words are more succinct (possibly even exponentially) than deterministic ones. In recent years, we see growing research on GFG automata, their theoretical properties, and applications [2,21,7,3,25,6,13].

Proving NP-hardness for DBW minimization, Schewe used a reduction from the vertex-cover problem [33]. Essentially[1], given an undirected graph $G = \langle V, E \rangle$, we seek a minimal DBW for the language S_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 $\{v_{i_j}, v_{i_{j+1}}\} \in E$. We can recognize S_G by an automaton obtained from G by replacing an edge $\{u_1, u_2\}$ by transitions from u_1 to u_2 and from u_2 to u_1, adding self loops to all vertices, labelling each transition by its destination, and requiring a run to traverse infinitely many transitions induced by 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 S_G. When, however, the acceptance condition is defined by a set of states, rather than transitions, we need to duplicate some states, and a minimal duplication corresponds to a minimal vertex cover. Consequently, finding a minimal DBW for S_G (or a minimal DCW for its complement) corresponds to finding a minimal vertex cover in G. Clearly, as a tDBW for S_G has the same structure as G, the reduction does not apply to tDBWs or tDCWs. On the other hand, it can be extended to GFG-NBWs and GFG-NCWs

[1] The reduction in [33] is more complicated and involves an additional letter. One of our contributions here is to show that the vertex-cover problem is NP-hard already for a class of graphs for which these complications are not required. Consequently, it is possible to simplify the reduction, and the description above is of the simplified version.

[34]. Interestingly, though, for GFG-tNCWs, minimization can be done in PTIME [1]. Thus, the complexity of minimization of tDBWs and tDCWs is still open, and so is the complexity for GFG-tNBWs.

Recall that for languages of finite words, a minimal DFW can be obtained in PTIME by merging of equivalent states. A similar algorithm is valid for determinisitic *weak* automata on infinite words: DBWs in which each strongly connected component is either contained in α or is disjoint from α [28,27]. In particular, *safety properties*, which assert that the system stays within some allowed region, can be recognized by DBWs in which all states are in α, which are a special case of weak automata. This raises the question of finding other natural fragments of ω-regular languages for which minimization could be solved in PTIME.

Of special interest in formal verification are *liveness* properties, which assert that something "good" eventually happens. For example, a process eventually enters its critical section or a grant is given infinitely often. We distinguish between three classes of liveness. Specifically, a language $L \subseteq \Sigma^\omega$ is Live1 if it has no "bad prefixes", thus every finite word in Σ^* can be extended to a word in L. Then, L is Live2 if $L = \Sigma^* \cdot L$, thus, every finite word in Σ^* can be extended by a word in L to a word in L, or, equivalently, every infinite word that has a suffix in L is also in L. Finally, L is Live3 if $L = \infty R$, for some language $R \subseteq \Sigma^*$ of finite words, thus L consists of words with infinitely many disjoint infixes in R. It is not hard to see that the classes are strictly ordered, in the sense that every Live3 language is Live2, every Live2 language is Live1, yet implication in the other direction does not hold. Also, the language S_G used for proving NP-hardness in [33] is not a liveness language. Indeed, finite sequences of vertices that do not correspond to a path in G are bad prefixes for S_G.

We study the minimization problem for automata recognizing languages in the three classes. We also consider the dual setting, where we minimize co-Büchi automata for languages that complement liveness languages. Note that while for deterministic automata, dualization of the acceptance condition complements the automaton, thus DBW and tDBW minimization coincides with DCW and tDCW minimization, this is not the case for GFG automata, where complementation should involve also a dualization of the nondeterministic branching mode (see [8] for a study of alternating GFG automata). In particular, as noted above, while minimization of GFG-tNCWs is in PTIME [1], the complexity for GFG-tNBWs is still open.

We first show that Schewe's reduction can be modified so that the vertex-cover problem is reduced to minimization of automata for Live1 languages. The main contribution in this part is a characterization of *nice* graphs: graphs for which the vertex-cover problem stays NP-hard yet they enjoy properties that enable us to significantly simplify the languages needed for the reduction. Then, given a nice graph G, we define a Live1 language L_G such that finding a minimal vertex-cover for G can be reduced to minimizing a DBW or a GFG-NBW for L_G, or to minimizing a DCW or a GFG-NCW for the complement of L_G. Essentially, while the language S_G does not allow prefixes that do not correspond to paths in G, the language L_G handles attempts to proceed to a vertex that is not connected by an edge the same way S_G handles self loops.

We continue to Live2 languages and describe a general scheme for transforming a language L to a Live2 language $L^\#$ such that minimization of automata for L can be reduced to minimization of automata for $L^\#$. Thus, minimization stays NP-hard for Live2 languages. Our scheme applies to deterministic and GFG automata, Büchi and

co-Büchi, with either state-based or transition-based acceptance. Consequently, while the problem of minimizing tDBWs and GFG-tNBWs is still open, our results imply that its complexity for general ω-regular languages coincides with its complexity for Live2 languages. Thus, efforts to find a PTIME algorithm can focus on the Live2 fragment.

Finally, we show that the transition from Live2 to Live3 languages is significant: while minimization of GFG-NCWs that recognize the complement of Live2 languages is NP-hard, minimization of GFG-NCWs that recognize the complement of Live3 languages can be done in PTIME. We find this result interesting, as we also show that Live3 languages maintain the combinatorial richness of GFG automata over deterministic ones. In particular, the exponential succinctness of GFG-NCWs over DCWs [21] applies also for languages whose complements are Live3, and the fact that GFG-NBWs need not be DBP [5] is exhibited also for Live3 languages. Also, for other classes of automata, the complexity of minimization of Live3 languages remains open.

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, and the set of finite words over Σ by Σ^*. For an index $i \geq 0$, we use $w[1, i]$ to denote the (possibly empty) prefix $\sigma_1 \cdot \sigma_2 \cdots \sigma_i$ of w. For $1 \leq i \leq j$, we use $w[i, j]$ to denote the infix $\sigma_i \cdot \sigma_{i+1} \cdots \sigma_j$ of w, and use $w[i, \infty]$ to denote the infinite suffix $\sigma_i \cdot \sigma_{i+1} \cdots$ of w. For a set A, we denote its complement by $\overline{A}$. In particular, if $R \subseteq \Sigma^*$ is a language over finite words and $L \subseteq \Sigma^\omega$ is language over infinite words, then $\overline{R} = \Sigma^* \setminus R$ and $\overline{L} = \Sigma^\omega \setminus L$.

2.1 Automata

A *nondeterministic automaton* on infinite words is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, 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 $s \in \delta(q, \sigma)$. If $|\delta(q, \sigma)| = 1$ for every state $q \in Q$ and letter $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*. The transition function δ can be viewed as 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 define the *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, as its number of states, thus, $|\mathcal{A}| = |Q|$.

Given an input word $w = \sigma_1 \cdot \sigma_2 \cdots$, a *run* of $\mathcal{A}$ on w 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 $r_{i+1} \in \delta(r_i, \sigma_{i+1})$, i.e, the run starts in the initial state and proceeds according to the transition function. 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 $\langle r_0, \sigma_1, r_1 \rangle, \langle r_1, \sigma_2, r_2 \rangle, \ldots \in \Delta^\omega$. We sometimes consider finite runs on finite words. In particular, 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 acceptance condition α determines which runs are "good". We consider *state-based* and *transition-based* automata. Let us start with *state-based* automata. Here, $\alpha \subseteq Q$, and we use the terms *α-states* and *$\overline{\alpha}$-states* to refer to states in α and in $Q \setminus \alpha$, respectively. For a run $r \in Q^\omega$, let $inf(r) \subseteq Q$ be the set of states that r visits infinitely often. Thus, $inf(r) = \{q : q = r_i \text{ for infinitely many } i\text{'s}\}$. In *Büchi* automata, r is *accepting* iff $inf(r) \cap \alpha \neq \emptyset$, thus if r visits states in α infinitely often. In *co-Büchi* automata, r is *accepting* iff $inf(r) \cap \alpha = \emptyset$, thus if r visits states in α only finitely often.

We proceed to *transition-based* automata. There, $\alpha \subseteq \Delta$ and acceptance depends on the set of transitions that are traversed infinitely often during the run. We use the terms *α-transitions* and *$\bar{\alpha}$-transitions* to refer to transitions in α and in $\Delta \setminus \alpha$, respectively. For a run $r \in \Delta^\omega$, we define $inf(r) = \{\langle q, \sigma, s\rangle \in \Delta : q = r_i, \sigma = \sigma_{i+1}, \text{ and } s = r_{i+1} \text{ for infinitely many } i\text{'s}\}$. As expected, in transition-based Büchi automata, r is accepting iff $inf(r) \cap \alpha \neq \emptyset$, and in transition-based co-Büchi automata, r is accepting iff $inf(r) \cap \alpha = \emptyset$. A run that is not accepting is *rejecting*. A word w is accepted by an automaton $\mathcal{A}$ 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.

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$. For a state $q \in Q$ of $\mathcal{A}$, we define $\mathcal{A}^q = \langle \Sigma, Q, q, \delta, \alpha\rangle$, i.e, $\mathcal{A}^q$ is the automaton obtained from $\mathcal{A}$ by setting the initial state to be q. We say that two states $q, s \in Q$ are *equivalent*, denoted $q \sim_{\mathcal{A}} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. We say that q *is reachable* if there is a finite word $x \in \Sigma^*$ with $q \in \delta(q_0, x)$, and say that q *is reachable from* s if q is reachable in $\mathcal{A}^s$.

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 hold:

1. The strategy f is consistent with the transition function. That is, $f(\epsilon) = q_0$, and for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $f(u \cdot \sigma) \in \delta(f(u), \sigma)$.
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 an accepting run of $\mathcal{A}$ on w.

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. Note that every deterministic automaton is GFG.

We use three-letter acronyms in $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{B}, \mathrm{C}\} \times \{\mathrm{W}\}$ to denote the different automata classes. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance condition type (Büchi or co-Büchi); and the third indicates that we consider automata on words. For transition-based automata, we start the acronyms with the letter "t", and for GFG automata, we write "GFG-" before the acronyms. For example, a GFG-NBW is a state-based GFG Büchi automaton, and a tDCW is a transition-based deterministic co-Büchi automaton.

For a class γ of automata, e.g., γ = GFG-NCW or γ = tDBW, we say that a language $L \subseteq \Sigma^\omega$ is γ-recognizable iff there is an automaton in the class γ that recognizes L. It is known [23,31] that GFG automata are as expressive as deterministic automata of the same acceptance condition, e.g., L is GFG-tNBW recognizable iff L is DBW recognizable.

For a class γ of automata, we say that a γ automaton $\mathcal{A}$ is *minimal* if for every equivalent γ automaton $\mathcal{B}$, namely one with $L(\mathcal{A}) = L(\mathcal{B})$, it holds that $|\mathcal{A}| \leq |\mathcal{B}|$.

In Appendix A.1, we define the notion of *semantic determinization* as well as *dualization* of automata on infinite words – notions that are used only in proofs in the appendix.

2.2 Liveness languages

For a language $R \subseteq \Sigma^*$ of finite words, we use ∞R to denote the language of infinite words that contain infinitely many disjoint infixes in R. Thus, $w \in \infty R$ iff there are infinitely many indices $i_1 \leq i'_1 < i_2 \leq i'_2 < \cdots$ such that $w[i_j, i'_j] \in R$, for all $j \geq 1$. Dually, $\neg\infty R$ is the language of infinite words that eventually contain only infixes in $\overline{R}$. Thus, there is an index i such that for all $j' \geq j \geq 0$ we have that $w[i+j, i+j'] \in \overline{R}$. Note that $\overline{\infty R} = \neg\infty R$.

Consider a language $L \subseteq \Sigma^\omega$. We say that a finite word $x \in \Sigma^*$ is a *bad prefix* for L if for all infinite words $y \in \Sigma^\omega$, we have that $x \cdot y \notin L$. Thus, there is no way to extend x to a word in L. Dually, x is a *good prefix* for L if all its extensions to an infinite word result in a word in L, thus $x \cdot y \in L$ for all $y \in \Sigma^\omega$.

We consider three levels of liveness. Consider a nonempty language $L \subseteq \Sigma^\omega$.

1. L is a Live1 language if it has no bad prefixes. Formally, for every finite word $x \in \Sigma^*$, there is an infinite word $y \in \Sigma^\omega$ such that $x \cdot y \in L$.
2. L is a Live2 language if $L = \Sigma^* \cdot L$. Formally, for every finite word $x \in \Sigma^*$, and infinite word $y \in L$, it holds that $x \cdot y \in L$. Equivalently, every word that has a suffix in L, is in L.
3. L is a Live3 language if $L = \infty R$, for some $R \subseteq \Sigma^*$. Thus, L consists of words with infinitely many disjoint infixes in R.

It is not hard to see that Live3 $\subseteq$ Live2 $\subseteq$ Live1. The other direction is not valid. For example, taking $\Sigma = \{a, b\}$, the language $L_1 = a \cdot \Sigma^\omega + \Sigma^* \cdot a \cdot a \cdot \Sigma^\omega$, of words that start with a or contain the infix $a \cdot a$, is Live1 and not Live2. Indeed, the word $b \cdot a \cdot b^\omega$ is in $(\Sigma^* \cdot L_1) \setminus L_1$. Then, $L_2 = \Sigma^* \cdot a \cdot a \cdot \Sigma^\omega$ is Live2 and not Live3.

Note that liveness languages need not be DBW-recognizable. For example, the Live1 and Live2 language $\Sigma^* \cdot b^\omega$ is not DBW-recognizable [24]. As for Live3 languages, if R is regular, then ∞R is DBW-recognizable [22].

We also consider languages that complement a liveness language, and say that a language L is a Doom1 language if $\overline{L}$ is a Live1 language, and similarly for Doom2 and Doom3 languages.

2.3 Graphs, nice graphs, and the vertex-cover problem

We consider undirected graphs $G = \langle V, E \rangle$, with a finite nonempty set V of vertices and a symmetric set $E \subseteq V \times V$ of edges. For simplicity, we assume that G has no loops or parallel edges. For a vertex $u \in V$, let $\eta(u)$ denote the set of *u's neighbors* in G; that is, $\eta(u) = \{v \in V : E(u, v)\}$. Then, $\overline{\eta(u)} = V \setminus \eta(u) = \{v \in V : \neg E(u, v)\}$. Note that since G has no self loops, then $u \in \overline{\eta(u)}$ for all vertices $u \in V$. For a vertex $u \in V$ and an edge $e = \langle x, y \rangle \in E$, we say that e is a *neighbor* of u if it at least one of its endpoints is a neighbor of u, thus $E(u, x)$ or $E(u, y)$.

For two vertices $u, v \in V$, we say that u and v are *separable* if there is an edge that is a neighbor of u but is not a neighbor of v, as well as an edge that is a neighbor of v but is not a neighbor of u. We say that G is *separable* if every two vertices $u, v \in V$ are separable. Then, G is *nice* if it is connected, separable, and for every vertex $u \in V$, there is a vertex v such that $v \neq u$ and $v \in \overline{\eta(u)}$. In particular, every nice graph has at least two vertices.

A set $C \subseteq V$ is a *vertex-cover* of G if each edge in G has at least one endpoint in C; that is, if $E(u, v)$, then $u \in C$ or $v \in C$. In the vertex-cover problem, we are given a graph G and an integer $k \geq 1$, and we have to decide whether G has a vertex-cover of size at most k.

In the rest of this section we prove that the vertex-cover problem is NP-hard already for nice graphs. For this, we analyze the reduction from 3SAT to the vertex-cover problem and argue that we can modify each 3CNF formula φ to an equivalent 3CNF formula φ' such that the graph that the reduction produces from φ' is nice.

Consider a 3CNF Boolean formula φ over the variable set $X = \{x_1, x_2, \ldots, x_n\}$. Let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of clauses in φ, with $c_j = (l_1^j \vee l_2^j \vee l_3^j)$. The standard reduction from satisfiability of 3CNF formulas to the vertex-cover problem generates, given φ, an undirected graph $G_\varphi = \langle V, E \rangle$ that consists of two types of "gadgets". For every variable $x_i \in X$, we have a variable-gadget consisting of two vertices x_i and $\overline{x_i}$, connected by an edge. Then, for each clause $c_j = (l_1 \vee l_2 \vee l_3)$, we have a clause-gadget, which is a triangle with three vertices, $c^j_{l_1}, c^j_{l_2}$, and $c^j_{l_3}$. Each vertex $c^j_{l_i}$ in the clause-gadget is connected by an edge to the literal l_i in its variable-gadget. It is not hard to see that the formula φ is satisfiable iff G_φ has a vertex-cover of size at most $n + 2m$. Indeed, each assignment induces a choice of one vertex from each variable-gadget, this choice covers the edges in the variable gadgets. In addition, we need two vertices to cover the edges in the triangles in the clause gadgets and we can choose these vertices in a way that also covers the edges between the vertex and clause gadgets iff the assignment is satisfying.

The graph G_φ need not be nice. For example, the graph G_φ for the formula $\varphi = (x_1 \vee x_2 \vee x_3)$, consists of three variable-gadgets connecting x_i and $\overline{x_i}$, and a single clause-gadget, and the set of edges that are neighbours of the vertex $\overline{x_i}$ is contained in the set of edges that are neighbours of the vertex x_i. Hence, G_φ is not separable.

Theorem 1. *The vertex-cover problem for nice graphs is NP-hard.*

Proof. Consider a 3CNF formula φ with m clauses over the variables $\{x_1, x_2, \ldots, x_n\}$. We assume that every clause in φ includes literals referring to three different variables. Note that otherwise, we can add variables and rewrite φ so that it satisfies this property. We define

$$\varphi' = \varphi \wedge \bigwedge_{1 \leq i \leq n} (x_i \vee z_1 \vee \neg z_2) \wedge (\neg x_i \vee \neg z_1 \vee z_2) \wedge (x_i \vee \neg z_1 \vee z_2) \wedge (\neg x_i \vee z_1 \vee \neg z_2),$$

where z_1 and z_2 are two new variables. It is not hard to see that φ is satisfiable iff φ' is satisfiable. Indeed, a satisfying assignment for φ can be extended to a satisfying assignemt for φ' by assigning True to z_1 and z_2. Conversely, as φ' implies φ, then a satisfying assignment for φ' also satisfies φ. Thus, φ is satisfiable iff the graph $G_{\varphi'}$ contains a vertex-cover of size at most $(n+2)+2(m+4n)$. In Appendix A.3, we prove that $G_{\varphi'}$ is nice. □

3 LIVE1 Languages

In this section we study the minimization problem for deterministic and GFG automata recognizing LIVE1 and DOOM1 languages. For a graph $G = \langle V, E \rangle$, we define the ω-regular language L_G over the alphabet V as the set of infinite words of the form

$$v_0 \cdot (\overline{\eta(v_0)})^* \cdot v_1 \cdot (\overline{\eta(v_1)})^* \cdot v_2 \cdot (\overline{\eta(v_2)})^* \cdots ,$$

where for all $i \geq 0$, it holds that $v_{i+1} \in \eta(v_i)$. Thus, a word $w \in L_G$ corresponds to an infinite path $v_0, v_1, v_2, \ldots$ in G, where each vertex v_i in the path contributes to w an infix in $v_i \cdot (\overline{\eta(v_i)})^*$, which is followed by the infix induced by v_{i+1}.

Note that every finite nonempty word $x = u_0 \cdot u_1 \cdot u_2 \cdots \in V^+$ induces a unique finite path in G: the path starts in u_0, and whenever it is in vertex v and the next letter is u_i, the path stays in v if $u_i \notin \eta(v)$ and proceeds to u_i if $u_i \in \eta(v)$. We say that the word x *leads to* vertex v if the path induced by x leads to v. In other words, x leads to v if $x \in v_0 \cdot (\overline{\eta(v_0)})^* \cdot v_1 \cdot (\overline{\eta(v_1)})^* \cdot v_2 \cdot (\overline{\eta(v_2)})^* \cdots v_k \cdot (\overline{\eta(v_k)})^*$, where $v_k = v$ and for all $0 \leq i \leq k-1$, it holds that $v_{i+1} \in \eta(v_i)$.

Example 1. Consider the graph G appearing in Fig. 1. A nonempty word of the form $(v_2 \cdot v_1)^*$ leads to v_1 and a word of the form $(v_2 \cdot v_1)^* \cdot v_2$ leads to v_2. Accordingly, the word $(v_2 \cdot v_1)^\omega$ is in L_G. All the words of the form $v_4 \cdot (v_2 \cdot v_1)^*$ or $v_4 \cdot (v_2 \cdot v_1)^* \cdot v_2$ lead to v_4, and so the word $v_4 \cdot (v_2 \cdot v_1)^\omega$ is not in L_G.

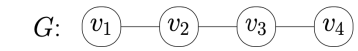

Fig. 1: The graph G.

Lemma 1. *For every nice graph $G = \langle V, E \rangle$, the language L_G is* LIVE1.

Proof. Consider a finite word $x \in V^+$, and assume that x leads to vertex v. Since G is nice, in particular connected and has at least two vertices, we get that there is a vertex $u \in \eta(v)$. Then, the word $x \cdot (u \cdot v)^\omega$ is in L_G. □

Note that L_G also has no good prefixes. Indeed, for every finite word x, the word $x \cdot u^\omega$ is not in L_G, for every vertex $u \in V$.

3.1 Minimizing automata for LIVE1 and DOOM1 languages

In this section we show that minimizing a GFG-NBW or a DBW that recognizes L_G, for a nice graph G, is NP-hard, and so is minimizing a GFG-NCW or a DCW that recognizes $\overline{L_G}$. We conclude that the minimization problem is NP-hard already for LIVE1 and DOOM1 languages.

We start by defining a DBW that recognizes L_G. Consider a nice graph $G = \langle V, E \rangle$. As has been the case with Schewe's language S_G, it is easy to define a tDBW for L_G from G by replacing each edge $\langle u_1, u_2 \rangle$ by a u_2-transition from u_1 to u_2, adding to each state v a self-loop labeled by all the letters in $\overline{\eta(v)}$, and requiring a run to traverse infinitely many transitions induced by edges of G. When considering DBWs, we define, for a subset of vertices $S \subseteq V$, the DBW $\mathcal{A}_{G,S} = \langle V, \{q_0\} \cup (V \times \{0\}) \cup (S \times \{1\}), q_0, \delta, \alpha \rangle$, where $\alpha = S \times \{1\}$, and δ is defined as follows (see Example 2).

- For all $u \in V$, we have $\delta(q_0, u) = \langle u, 0\rangle$.
- For all $u \in V$ and $v \in \eta(u)$, we have $\delta(\langle u, 0\rangle, v) = \langle u, 0\rangle$. In addition, if $u \in S$, then $\delta(\langle u, 1\rangle, v) = \langle u, 0\rangle$.
- For all $u \in V$ and $v \in \eta(u)$, if $v \in S$, then $\delta(\langle u, 0\rangle, v) = \langle v, 1\rangle$, and if $v \notin S$, then $\delta(\langle u, 0\rangle, v) = \langle v, 0\rangle$. In addition, if $u \in S$, then $\delta(\langle u, 1\rangle, v) = \delta(\langle u, 0\rangle, v)$.

Thus, for every vertex $v \notin S$, the DBW $\mathcal{A}_{G,S}$ includes a "0-state" $\langle v, 0\rangle$, and for every vertex $v \in S$, it includes both a "0-state" $\langle v, 0\rangle$ and a "1-state" $\langle v, 1\rangle$. The 0-state $\langle v, 0\rangle$ has a self-loop labeled by letters in $\eta(v)$. Reading a letter $u \in \eta(v)$ from $\langle v, 0\rangle$ or from $\langle v, 1\rangle$, the DBW moves to $\langle u, 0\rangle$ if $u \notin S$ and moves to $\langle u, 1\rangle$ if $u \in S$. The 1-state $\langle v, 1\rangle$ does not have a self loop, and rather it moves to $\langle v, 0\rangle$ with letters in $\eta(v)$. As $\alpha = S \times \{1\}$, the run of $\mathcal{A}_{G,S}$ is accepting iff the path induced by the input word traverses infinitely many edges with endpoints in S. Hence, $\mathcal{A}_{G,S}$ captures infinitely many traversals of edges of G iff the set S is a vertex-cover of G. Formally, we have the following (see proof in Appendix A.4).

Lemma 2. *$L(\mathcal{A}_{G,S}) \subseteq L_G$, and $L_G \subseteq L(\mathcal{A}_{G,S})$ iff S is a vertex-cover of G.*

Example 2. Consider the graph $G = \langle\{v_1, v_2, v_3\}, E\rangle$, appearing in Fig. 2. Note that G is not nice, yet L_G is Live1. For the set $S = \{v_1, v_3\}$, the DBW $\mathcal{A}_{G,S}$ appears in Fig. 3.

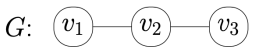

Fig. 2: The graph G.

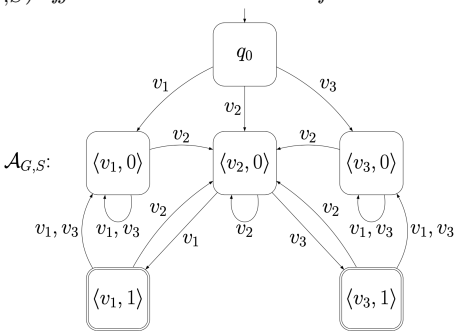

Fig. 3: The DBW $\mathcal{A}_{G,S}$.

While $\mathcal{A}_{G,S}$, for a minimal vertex cover S, is a natural candidate for a minimal DBW that recognizes L_G, a general result about minimization should consider arbitrary DBWs and GFG-NBWs for the language. We continue and specify properties of such automata. Consider a nice graph $G = \langle V, E\rangle$. Let $\mathcal{A} = \langle V, Q, q_0, \delta, \alpha\rangle$ be a GFG-NBW that recognizes the language L_G. For a vertex $v \in V$ and a state $q \in Q$, we say that q is a *v-state* if q is reachable from q_0 upon reading a finite nonempty word that leads to v. In particular, all the v-successors of q_0 are v-states.

Proposition 1. *For every GFG-NBW $\mathcal{A} = \langle V, Q, q_0, \delta, \alpha\rangle$ that recognizes the language L_G, the following hold.*

1. *For every vertex $v \in V$, all the v-states are equivalent.*
2. *For every two vertices $u \neq v \in V$, the u-states are not equivalent to the v-states.*
3. *The initial state q_0 is not a v-state, for all vertices $v \in V$.*
4. *$|\{q \in Q \setminus \alpha : q$ is a v-state for some vertex $v \in V\}| \geq |V|$; that is, the number of v-states not in α is at least $|V|$.*
5. *For every edge $\langle u, v\rangle \in E$, there is a v-state or a u-state in α.*

Using the properties in Proposition 1 (see proof in Appendix A.5), we can argue that a nice graph G has a vertex cover of size at most $k \geq 1$ iff the DBW $\mathcal{A}_{G,V}$, which recognizes L_G, has an equivalent GFG-NBW or DBW with at most $|V| + k + 1$ states. Hence, we can conclude with the following (see proof in Appendix A.6).

Theorem 2. *The minimization problem is NP-hard already for DBWs and GFG-NBWs that recognize* Live1 *languages.*

We continue to DCWs and GFG-NCWs that recognize Doom1 languages. For DCWs, NP-hardness for minimization follows immediately from Theorem 2. For GFG-NCWs, we have to carefully examine the proof of Proposition 1 and see that it stays valid in the dual setting. In Appendix A.7 we describe the proposition and its proof for the co-Büchi case, as well as its use in proving the following, namely the co-Büchi counterpart of Theorem 2.

Theorem 3. *The minimization problem is NP-hard already for DCWs and GFG-NCWs that recognize* Doom1 *languages.*

4 Live2 Languages

In this section, we study minimization for Live2 and Doom2 languages. Note that the language L_G used in Section 3 is not Live2. To see this, consider the graph G from Example 1. Note that the word $w = v_4 \cdot (v_2 \cdot v_1)^\omega \in V^* \cdot L_G$ has the suffix $(v_2 \cdot v_1)^\omega \in L_G$, yet $w \notin L_G$. In fact, for every nice graph G, we have that L_G is not Live2. Indeed, every nice graph G has at least two vertices u and v. Then, as u and v are separable, there is an edge $\langle u_1, u_2 \rangle \in E$ that is a neighbor of u but not a neighbor of v. The word $w = v \cdot (u_1 \cdot u_2)^\omega \in V^* \cdot L_G$ has the suffix $(u_1 \cdot u_2)^\omega \in L_G$, yet $w \notin L_G$.

4.1 Minimizing DBWs and GFG-NBWs for Live2 languages

In this section, we show that minimizing DBWs and GFG-NBWs that recognize Live2 languages is NP-hard.

The idea behind our proof is as follows. We define an operation on ω-regular languages, such that for every language $L \subseteq \Sigma^\omega$, and letter $\# \notin \Sigma$, the operation constructs a language $L^\# \subseteq (\Sigma \cup \{\#\})^\omega$ that is Live2. Then, we define two operations on automata. The first is applied to a DBW $\mathcal{A}$ over the alphabet Σ and it constructs a DBW $\mathcal{A}^\#$ over the alphabet $\Sigma \cup \{\#\}$, such that $L(\mathcal{A}^\#) = L(\mathcal{A})^\#$. The second operation is applied to a GFG-NBW $\mathcal{A}^\#$ over the alphabet $\Sigma \cup \{\#\}$, it constructs a GFG-NBW $\mathcal{A}$ over the alphabet Σ, and if $L(\mathcal{A}^\#) = L^\#$ for some language $L \subseteq \Sigma^\omega$, then $L(\mathcal{A}) = L$. Moreover, the construction preserves determinization. The blow-up in both constructions is such that when we take L to be the language L_G from Section 3.1, we can replace the reduction there by a reduction that seeks minimal automata for the language $L_G^\#$.

We start by describing the operation on ω-regular languages. For $L \subseteq \Sigma^\omega$, let

$$L^\# = ((\Sigma \cup \{\#\})^* \cdot \# \cdot L) \cup (\infty\#).$$

Thus, $L^\#$ is defined over the alphabet $\Sigma \cup \{\#\}$, and it consists of all infinite words that either have a suffix of the form $\# \cdot L$ or contain infinitely many #'s. It is not hard to see that $L^\# = (\Sigma \cup \{\#\})^* \cdot L^\#$, thus $L^\#$ is Live2.

We continue with the operations on automata.

Theorem 4. *Given a DBW $\mathcal{A}$ over Σ such that the initial state of $\mathcal{A}$ has no incoming transitions, we can construct a DBW $\mathcal{A}^\#$ over $\Sigma \cup \{\#\}$ such that the following hold.*

1. $L(\mathcal{A}^{\#}) = L(\mathcal{A})^{\#}$.
2. *If* $L(\mathcal{A})$ *is* Live1*, then* $|\mathcal{A}^{\#}| = |\mathcal{A}| + 1$*. Otherwise,* $|\mathcal{A}^{\#}| = |\mathcal{A}|$.

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. If $L(\mathcal{A})$ is Live1, then we define $\mathcal{A}^{\#} = \langle \Sigma \cup \{\#\}, Q \cup \{q^0_{\#}\}, q^0_{\#}, \delta_{\#}, \alpha_{\#} \rangle$, for $q^0_{\#} \notin Q$, and a transition function $\delta_{\#}$ defined as follows (see an example in Appendix A.9). For every $s \in Q$ and $\sigma \in \Sigma$, we have $\delta_{\#}(s, \sigma) = \delta(s, \sigma)$ and $\delta_{\#}(s, \#) = q_0$. Then, for the state $q^0_{\#}$, we have that $\delta(q^0_{\#}, \sigma) = q^0_{\#}$ for all $\sigma \in \Sigma$, and $\delta(q^0_{\#}, \#) = q_0$. Thus, $\mathcal{A}^{\#}$ is obtained from $\mathcal{A}$ by adding a new state $q^0_{\#}$ that has a Σ-labeled self-loop that goes with # to the state q_0, to which we move upon reading # from all states. In addition, $\alpha_{\#} = \alpha \cup \{q_0\}$.

If $L(\mathcal{A})$ is not Live1, then $L(\mathcal{A})$ has at least one bad prefix, and so $\mathcal{A}$ must contain a state q with $L(\mathcal{A}^q) = \emptyset$. We assume w.l.o.g that q is a rejecting sink in $\mathcal{A}$, and define $\mathcal{A}^{\#}$ as above, except that instead of defining the state $q^0_{\#}$ as a new state, we define $q^0_{\#} = q$. Note that also in this case, the initial state $q^0_{\#}$ has a Σ-labeled self-loop and a #-transition to q_0.

In Appendix A.8, we prove that $\mathcal{A}^{\#}$ satisfies the two properties. □

We continue to the reverse operation, where we start with a GFG-NBW.

Theorem 5. *Given a GFG-NBW* $\mathcal{A}^{\#}$ *over* $\Sigma \cup \{\#\}$ *such that* $L(\mathcal{A}^{\#}) = L^{\#}$ *for some* $L \subseteq \Sigma^{\omega}$*, we can construct a GFG-NBW* $\mathcal{A}$ *over* Σ*, such that the following hold.*

1. $\mathcal{A}$ *recognizes* L.
2. *If* $\mathcal{A}^{\#}$ *is determinstic, then so is* $\mathcal{A}$.
3. *If* L *is* Live1*, then* $|\mathcal{A}| \leq |\mathcal{A}^{\#}| - 1$*. Otherwise,* $|\mathcal{A}| \leq |\mathcal{A}^{\#}|$.

Proof. Let $\mathcal{A}^{\#} = \langle \Sigma \cup \{\#\}, Q, q_0, \delta, \alpha \rangle$, and let $f : (\Sigma \cup \{\#\})^* \to Q$ be a strategy witnessing the GFGness of $\mathcal{A}^{\#}$. We obtain $\mathcal{A}$ from $\mathcal{A}^{\#}$ by removing all #-transitions and choosing $q_{\#} = f(\#)$ to be its initial state. Clearly, $\mathcal{A}$ is over Σ and determinization of $\mathcal{A}^{\#}$ is preserved in $\mathcal{A}$. In Appendix A.10, we prove that $\mathcal{A}$ is GFG and recognizes L. Moreover, if L is Live1, then we can remove a state from $\mathcal{A}$ and get an equivalent automaton, resulting in an automaton with at most $|\mathcal{A}^{\#}| - 1$ states. Essentially, the strategy $g : \Sigma^* \to Q$ that witnesses the GFGness of $\mathcal{A}$ is such that for all $x \in \Sigma^*$, we have $g(x) = f(\# \cdot x)$. □

We can now use the operations in order to extend the reduction from Section 3.1 to Live2 languages. Essentially (see proof in Appendix A.11), we show that a nice graph $G = \langle V, E \rangle$ has a vertex cover of size at most $k \geq 1$ iff the DBW $\mathcal{A}^{\#}_{G,V}$, namely the DBW obtained by applying the construction from Theorem 4 on the DBW $\mathcal{A}_{G,V}$ from Section 3.1, has an equivalent GFG-NBW or DBW with at most $|V| + k + 2$ states.

Theorem 6. *The minimization problem is NP-hard for DBWs and GFG-NBWs that recognize* Live2 *languages.*

4.2 Minimizing DCWs and GFG-NCWs for Doom2 languages

We continue to co-Büchi automata for Doom2 langauges. As has been the case with Live1 languages, Theorem 6 implies that the minimization problem for DCWs that

recognize Doom2 languages is NP-hard, yet does not imply hardness of minimizing GFG-NCWs that recognize Doom2 languages. Also, while Theorem 4 considers DBWs and can be dualized, Theorem 5 considers GFG-NBWs, and thus its dualization requires a proof (see Appendix A.12):

Theorem 7. *Given a GFG-NCW $\mathcal{A}^\#$ over $\Sigma \cup \{\#\}$ such that $L(\mathcal{A}^\#) = \overline{L^\#}$ for some $L \subseteq \Sigma^\omega$, we can construct a GFG-NCW $\mathcal{A}$ over Σ such that the following hold.*

1. *$\mathcal{A}$ recognizes $\overline{L}$.*
2. *If $\mathcal{A}^\#$ is determinstic, then so is $\mathcal{A}$.*
3. *If $\overline{L}$ is* Doom1*, then $|\mathcal{A}| \leq |\mathcal{A}^\#| - 1$. Otherwise, $|\mathcal{A}| \leq |\mathcal{A}^\#|$.*

We can now use a reduction similar to the one in Theorem 6 and prove that a nice graph G has a vertex-cover of size at most $k \geq 1$ iff the DCW $\widetilde{\mathcal{A}}^\#_{G,V}$, which dualizes $\mathcal{A}^\#_{G,V}$, has an equivalent GFG-NCW of size at most $|V| + k + 2$. See proof in Appendix A.13.

Theorem 8. *The minimization problem is NP-hard for DCWs and GFG-NCWs that recognize* Doom2 *languages.*

4.3 Minimizing Automata With Transition-Based Acceptance for Live2 Languages

In this section, we show that minimizing GFG-tNBWs or tDBWs recognizing Live2 languages is not easier than minimizing GFG-tNBWs or tDBWs, respectively, recognizing general languages. Thus, while the complexity of minimization of tDBWs and GFG-tNBWs is still open, it is sufficient to restrict attention to transition-based Büchi automata recognizing Live2 languages.

The idea of our proof is to modify the constructions used in Section 4.1 to GFG-tNBWs. Note that while Theorem 5 already considers GFG-NBWs, and thus the extension only has to adjust the type of acceptance, which is easy, Theorem 4 considers deterministic automata, and so the extension is more involved. In particular, one has to show how a function that witnesses the GFGness of $\mathcal{A}$ induces a strategy that witnesses the GFGness of $\mathcal{A}^\#$. Formally, we have the following (see proof in Appendix A.14).

Theorem 9. *Given a GFG-tNBW $\mathcal{A}$ over Σ, we can construct a GFG-tNBW $\mathcal{A}^\#$ over $\Sigma \cup \{\#\}$ such that the following hold.*

1. *$L(\mathcal{A}^\#) = L(\mathcal{A})^\#$.*
2. *If $\mathcal{A}$ is determinsitic, then so is $\mathcal{A}^\#$.*
3. *If $L(\mathcal{A})$ is* Live1*, then $|\mathcal{A}^\#| = |\mathcal{A}| + 1$. Otherwise, $|\mathcal{A}^\#| = |\mathcal{A}|$.*

We can now reduce minimization of a general GFG-tNBW to minimization of a GFG-tNBW for a Live2 language as follows. Given a GFG-tNBW $\mathcal{A}$ and an integer $k \geq 1$, our reduction returns the GFG-tNBW $\mathcal{A}^\#$ and the integer k' such that $k' = k + 1$ if $L(\mathcal{A})$ is Live1, and $k' = k$ otherwise (note that the latter condition can be checked in polynomial time). In Appendix A.15 we prove that $\mathcal{A}$ has an equivalent GFG-tNBW of size at most k iff $\mathcal{A}^\#$ has an equivalent GFG-tNBW of size at most k', and similarly for tDBWs. Hence, we can conclude with the following.

Theorem 10. *Minimizing GFG-tNBWs or tDBWs recognizing* Live2 *languages is not easier than minimizing general GFG-tNBWs or tDBWs, respectively.*

We continue to tDCWs and show that the reduction from Theorem 10 implies that minimizing tDCWs recognizing Doom2 languages is not easier than minimizing tDCWs recognizing general languages. Hence, we can conclude with the following (see proof in Appendix A.16):

Theorem 11. *Minimizing tDCWs recognizing* Doom2 *languages is not easier than minimizing general tDCWs.*

5 Live3 Languages

Recall that minimization of GFG-NCWs is NP-hard [34]. Moreover, by Theorem 8, NP-hardness applies already to Doom2 languages. Our main result in this section is that the transition from Live2 to Live3 languages is significant: minimization of GFG-NCWs that recognize Doom3 languages can be done in PTIME. We first argue that our result is surprising, in the sense that Live3 and Doom3 languages maintain the combinatorial richness of GFG automata. Specifically, recall that GFG-NBWs and GFG-NCWs may not be determinizable by pruning (DBP), and GFG-NCWs may be exponentially more succinct than DCWs [5,21,19]. We show that these advantages of GFG automata are valid already for Live3 and Doom3 languages. For the co-Büchi case, the languages used in [21,19] are already Doom3. For the Büchi case, our example is new.

Theorem 12. *There are GFG-NBWs and GFG-NCWs for* Live3 *and* Doom3 *languages that are not DBP. Moreover, GFG-NCWs for* Doom3 *languages may be exponentially more succinct than DCWs.*

Proof. It was shown [21,19] that GFG-NCWs may be exponentially more succinct than DCWs. It is not hard to see that the languages used there are Doom3. The latter implies also that GFG-NCWs recognizing Doom3 languages need not be DBP. Also, while the results there are for automata with transition-based acceptance, they apply also in the state-based setting. Indeed, transforming an NCW to an equivalent tNCW involves no blow-up, and transforming a tNCW to an NCW at most doubles the state-space. Both transformation preserve determinism, GFGness, and DBPness.

For Live3 languages and GFG-tNBWs, consider the tNBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ appearing in Fig. 4 (in the figure, dashed transitions are α-transitions). Let $R_x = a^+ \cdot x \cdot (x + y)^*$, $R_y = a^+ \cdot y \cdot (x+y)^*$, and $R = (R_x \cdot R_x + R_y \cdot R_y)$. In Appendix A.17, we prove that $\mathcal{A}$ recognizes the Live3 language ∞R, is GFG, yet not DBP. In addition, we show that by duplicating the state space of a GFG-tNBW $\mathcal{A}$ we can obtain an equivalent GFG-NBW $\mathcal{A}'$ such that $\mathcal{A}$ is DBP iff $\mathcal{A}'$ is DBP. Thus, the example applies also to GFG-NBWs. □

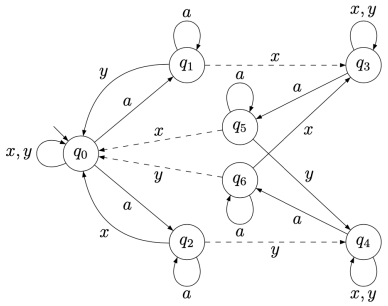

Fig. 4: The GFG-tNBW $\mathcal{A}$ for ∞R.

We continue and show that minimizing GFG-NCWs that recognize Doom3 languages can in done in PTIME. The main idea behind the minimization process is a construction that given a GFG-tNCW for a Doom3 language L, returns a GFG-NCW for L with one additional state, along with an observation that for every (non-trivial) Doom3 language L, a minimal GFG-NCW for L must be strictly bigger than a minimal GFG-tNCW for L. Thus, it is possible to use minimization of GFG-tNCWs, which can be done in PTIME [1].

Theorem 13. *The minimization problem for GFG-NCWs recognizing* Doom3 *languages can be solved in PTIME.*

Proof. Consider a GFG-NCW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ that recognizes a language of the form $\neg\infty R$, for some $R \subseteq \Sigma^*$.

Consider the GFG-NCW $\mathcal{B}$ that is obtained from $\mathcal{A}$ as follows (see Example 3).

1. Let $\mathcal{U} = \langle \Sigma, Q_\mathcal{U}, q^0_\mathcal{U}, \Delta_\mathcal{U}, \alpha_\mathcal{U} \rangle$ be the GFG-tNCW obtained by applying the minimization algorithm of [1] on the GFG-tNCW obtained from $\mathcal{A}$ by defining all the transitions that leave an α-state as α-transitions.
2. If $L(\mathcal{A})$ is trivial, then $\mathcal{B}$ can be defined as a one state DCW that is equivalent to $\mathcal{A}$. Otherwise, $\mathcal{B} = \langle \Sigma, Q_\mathcal{U} \cup \{s\}, q^0_\mathcal{U}, \Delta_\mathcal{B}, \alpha_\mathcal{B} \rangle$, where $\alpha_\mathcal{B} = \{s\}$ and $\Delta_\mathcal{B} = (\Delta_\mathcal{U} \setminus \alpha_\mathcal{U}) \cup \{\langle q, \sigma, s\rangle : \text{ there exists } q' \in Q_\mathcal{U} \text{ such that } \langle q, \sigma, q' \rangle \in \alpha_\mathcal{U}\} \cup (\{s\} \times \Sigma \times Q_\mathcal{U})$. Thus, $\mathcal{B}$ is obtained from $\mathcal{U}$ by adding the new state s, which is the only state in $\alpha_\mathcal{B}$, redirecting all the $\alpha_\mathcal{U}$-transitions of $\mathcal{U}$ to s, and adding σ-transitions from s to all the states of $\mathcal{U}$, for all $\sigma \in \Sigma$.

In Appendix A.18 we show that after applying a simple preprocessing step, we can modify $\mathcal{A}$ so that it satisfies some simple syntactic and semantic properties. Then, we prove that $L(\mathcal{B}) = L(\mathcal{A})$. Essentially, the equivalence follows from the following properties of the minimization algorithm in [1]:

1. If all the states of $\mathcal{A}$ are equivalent (which is the case in GFG-NCWs for Doom3 languages), then all the states of $\mathcal{U}$ are equivalent.
2. The GFG-tNCW $\mathcal{U}$ is *safe-homogenous*: for every state $q \in Q_\mathcal{U}$ and letter $\sigma \in \Sigma$, all the σ-labeled transitions from q are in $\alpha_\mathcal{U}$, or they are all not in $\alpha_\mathcal{U}$.
3. The GFG-tNCW $\mathcal{U}$ is *safe-deterministic*: for every state $q \in Q_\mathcal{U}$ and letter $\sigma \in \Sigma$, there is at most one σ-labeled transition that leaves q and is not in $\alpha_\mathcal{U}$.

Finally, as we detail in Appendix A.18, a minimal GFG-tNCW $\mathcal{U}$ recognizing a non-trivial Doom3 language, cannot have states such that all cycles through them traverse transitions in $\alpha_\mathcal{U}$. We show that this implies that a minimal equivalent GFG-NCW must be strictly bigger than a minimal GFG-tNCW, which implies the minimality of $\mathcal{B}$. □

Example 3. Consider the tNCW $\mathcal{U}$ in Fig. 5. The dashed transitions are α-transitions. The tNCW $\mathcal{U}$ recognizes the Doom3 language $L = \neg\infty(a \cdot (a + b)^* \cdot b)$, consisting of words with finitely many a's or finitely many b's. It is not hard to see that by pruning out the α-self-loops, we get an equivalent tDCW, thus $\mathcal{U}$ is DBP, in particular GFG. In addition, as there is no GFG-tNCW recognizing L with a single state, then $\mathcal{U}$ is a minimal GFG-tNCW. The GFG-NCW $\mathcal{B}$ is obtained form $\mathcal{U}$ by adding a new state s, directing the α-transitions to s, and nondeterministically branching from s to all states, with all letters.

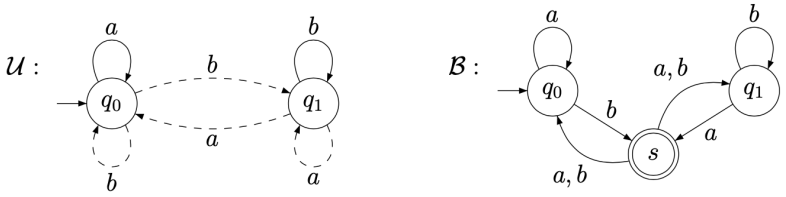

Fig. 5: The construction of the GFG-NCW $\mathcal{B}$ from $\mathcal{U}$.

References

1. B. Abu Radi and O. Kupferman. Minimizing GFG transition-based automata. In *Proc. 46th ICALP*, volume 132 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
2. B. Abu Radi, O. Kupferman, and O. Leshkowitz. A hierarchy of nondeterminism. In *46th MFCS*, volume 202 of *LIPIcs*, pages 85:1–85:21, 2021.
3. M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th FST & TCS*, volume 122 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
4. R. Bloem, K. Chatterjee, and B. Jobstmann. Graph games and reactive synthesis. In *Handbook of Model Checking.*, pages 921–962. Springer, 2018.
5. 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, pages 89–100, 2013.
6. U. Boker, D. Kuperberg, K. Lehtinen, and M. Skrzypczak. On the succinctness of alternating parity good-for-games automata. In *Proc. 40th FST & TCS*, volume 182 of *LIPIcs*, pages 41:1–41:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
7. U. Boker, O. Kupferman, and M. Skrzypczak. How deterministic are Good-For-Games automata? In *Proc. 37th FST & TCS*, volume 93 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, 2017.
8. U. Boker and K. Lehtinen. Good for games automata: From nondeterminism to alternation. In *Proc. 30th CONCUR*, volume 140 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
9. 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.
10. Th. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th ICALP*, LNCS 5556, pages 139–150. Springer, 2009.
11. 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 ATVA*, LNCS 9938, pages 122–129. Springer, 2016.
12. J. Esparza, O. Kupferman, and M.Y. Vardi. Verification. In *Handbook AutoMathA*, pages 549–588. European Mathematical Society, 2018.
13. R. Faran and O. Kupferman. On (I/O)-aware good-for-games automata. In *18th ATVA*, LNCS 12302, pages 161–178. Springer, 2020.
14. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proc. 13th CAV*, LNCS 2102, pages 53–65. Springer, 2001.
15. 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*, LNCS 2529, pages 308–326. Springer, 2002.

16. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.
17. T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th CSL*, LNCS 4207, pages 394–410. Springer, 2006.
18. 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.
19. S. Iosti and D. Kuperberg. Eventually safe languages. In *23rd International Conference on Developments in Language Theory*, LNCS 11647, pages 192–205, 2019.
20. T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.
21. D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Proc. 42nd ICALP*, pages 299–310, 2015.
22. O. Kupferman and O. Leshkowitz. On repetition languages. In *45th MFCS*, Leibniz International Proceedings in Informatics (LIPIcs), 2020.
23. O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.
24. L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.
25. K. Lehtinen and M. Zimmermann. Good-for-games ω-pushdown automata. In *Proc. 35th LICS*, 2020.
26. W. Li, Sh. Kan, and Z. Huang. A better translation from LTL to transition-based generalized Büchi automata. *IEEE Access*, 5:27081–27090, 2017.
27. C. Löding. Efficient minimization of deterministic weak ω-automata. *Information Processing Letters*, 79(3):105–109, 2001.
28. 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 LICS*, pages 422–427, 1988.
29. J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.
30. A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.
31. D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th STACS*, LNCS 1373. Springer, 1998.
32. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.
33. S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th FST & TCS*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.
34. S. Schewe. Minimising Good-For-Games automata is NP-complete. In *Proc. 40th FST & TCS*, volume 182 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
35. S. Sickert, J. Esparza, S. Jaax, and J. Křetínský. Limit-deterministic Büchi automata for linear temporal logic. In *Proc. 28th CAV*, LNCS 9780, pages 312–332. Springer, 2016.
36. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

A Proofs

A.1 Semantic Determinizm

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We say that $\mathcal{A}$ is *semantically deterministic* (SD, for short) 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. Formally, if $s, s' \in \delta(q, \sigma)$, then $s \sim_{\mathcal{A}} s'$.

The following proposition follows immediately from the definitions and is useful in some of our proofs.

Proposition 2. *Consider an SD automaton $\mathcal{A}$, and states $q, s \in Q$. If $q \sim_{\mathcal{A}} s$, then for every $\sigma \in \Sigma$, $q' \in \delta(q, \sigma)$, and $s' \in \delta(s, \sigma)$, we have that $q' \sim_{\mathcal{A}} s'$.*

For a deterministic Büchi automaton $\mathcal{A}$, we use $\widetilde{\mathcal{A}}$ to denote the deterministic co-Büchi automaton obtained by viewing $\mathcal{A}$ as a co-Büchi automaton. The Büchi and co-Büchi acceptance conditions are *dual* in the sense that $L(\mathcal{A}) = \overline{L(\widetilde{\mathcal{A}})}$. Similarly, for a deterministic co-Büchi automaton $\mathcal{A}$, we let $\widetilde{\mathcal{A}}$ denote the dual deterministic Büchi automaton, which complements $\mathcal{A}$.

A.2 Semantic determinization of GFG automata

Proposition 3. *Consider a GFG automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We can, in polynomial time, transform $\mathcal{A}$ into an equivalent GFG automaton that is SD, of the same class, and without increasing its state-space.*

Proof. For an automaton $\mathcal{A}$, 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 $f(u \cdot \sigma) = q'$. For all automata classes that we consider, states that are not GFG can be detected in polynomial time [21,3][2], and as all states that are used by a strategy that witnesses $\mathcal{A}$'s GFGness are GFG, the removal of non-GFG states does not affect $\mathcal{A}$'s language nor its GFGness. Note that removing the non-GFG states may result in an automaton with a non-total transition function, in which case we add a rejecting sink. Note that we add a rejecting sink only after removing some state, thus this step does not add states. Now, using the fact that language containment of GFG Büchi and GFG co-Büchi automata can be checked in polynomial time [16,21], and transitions that are used by strategies are covering [21], one can semantically determinize $\mathcal{A}$ by removing non-covering transitions. □

[2] We note that some of the results were shown only for state-based automata or only for transition-based automata, but it is okay to use them for all automata classes that we consider. Indeed, given a state-based automaton $\mathcal{A}$, one can turn it to a transition-based by considering transitions that leave α-states as α transitions. Conversely, if $\mathcal{A}$ is a transition-based automaton with the state-space Q, then one can turn it into a state-based automaton by duplicating its state-space to $Q \times \{0, 1\}$, where the right coordinate indicates whether we have just traversed an α-transition in $\mathcal{A}$. As both transformations preserve language and GFGness, then deciding GFGness is easy for all automata classes.

A.3 Details of the Proof of Theorem 1

First, note that $G_{\varphi'}$ is connected. Indeed, for every variable x_i of φ, the clause-gadget corresponding to the clause $(x_i \vee z_1 \vee \neg z_2)$ in φ' connects the variable gadgets corresponding to x_i, z_1 and z_2. Therefore, there is a path p that traverses all the variable gadgets in $G_{\varphi'}$. Finally, as every clause-gadget is connected to at least one variable-gadget, it follows that $G_{\varphi'}$ is connected as one can take detours from the path p to traverse also every clause-gadget.

Recall that every clause in φ includes literals referring to three different variables. It is easy to see that this property is maintained in φ'. In addition, note that every literal in φ' appears in at least two clauses. Finally, we show that every 3CNF formula θ over $X = \{x_1, x_2, \ldots, x_n\}$ with clauses $C = \{c_1, c_2, \ldots, c_m\}$ such that every clause includes literals referring to three different variables and every literal appears in at least two clauses, is such that 1) G_θ is separable, and 2) for every[3] vertex u, there is a vertex v such that $v \neq u$ and $v \in \overline{\eta(u)}$. Since φ' satisfy these conditions, we are done.

Let $G_\theta = \langle V, E \rangle$. Consider distinct vertices $u, v \in V$. We show that u and v are separable. We distinguish between several cases:

- u and v are vertices of the same variable-gadget: w.l.o.g $u = x_i$ and $v = \overline{x_i}$, for some variable $x_i \in X$. As x_i and $\overline{x_i}$ do not participate in the same clause and as every literal appears in some clause, then there is a clause $c_j = (x_i \vee l_2 \vee l_3) \in C$ that contains x_i but does not contain $\overline{x_i}$. By the definition of G_θ, u is connected to the clause-gadget of c_j and v is not, in particular, the edge $\langle c^j_{x_i}, c^j_{l_2} \rangle$ is a neighbor edge of u but not a neighbor edge of v. Similarly, one can consider a clause that contains $\overline{x_i}$ and does not contain x_i and define an edge that is a neighbor of v but not a neighbor of u.
- u and v are vertices of different variable-gadgets: w.l.o.g $u = x_i$ and $v = x_j$, for some distinct variables x_i and x_j. By the definition of G_θ there are no edges connecting different variable-gadgets, and hence the edge $\langle x_i, \overline{x_i} \rangle$ is a neighbor edge of u but not a neighbor edge of v. Similarly, the edge $\langle x_j, \overline{x_j} \rangle$ is a neighbor edge of v but not a neighbor edge of u.
- u and v are vertices of the same clause-gadget: w.l.o.g $u = c^j_{x_i}$ and $v = c^j_{x_k}$, for some distinct variables $x_i, x_k \in X$ and a clause $c_j = (x_i \vee x_k \vee l_3) \in C$. By the definition of G_θ, $c^j_{x_k}$ is not connected to x_i or to $\overline{x_i}$. Hence the edge $\langle x_i, \overline{x_i} \rangle$ is a neighbor edge of u but not a neighbor edge of v. Similarly, the edge $\langle x_k, \overline{x_k} \rangle$ is a neighbor edge of v but not a neighbor edge of u.
- u and v are vertices of different clause gadgets: w.l.o.g $u = c^{j_1}_{x_{i_1}}$ and $v = c^{j_2}_{x_{i_2}}$, for some distinct clauses $c_{j_1} = (x_{i_1} \vee l_2 \vee l_3), c_{j_2} \in C$. By the definition of G_θ, there are no edges connecting different clause gadgets. Hence the edge $\langle c^{j_1}_{x_{i_1}}, c^{j_1}_{l_2} \rangle$ is a neighbor edge of u but not a neighbor edge of v. Similarly, any edge in the clause-gadget of c_{j_2} is a neighbor edge of v but not a neighbor edge of u.
- u is a vertex of a variable-gadget and v is a vertex of a clause-gadget: w.l.o.g $u = x_i$, for some variable x_i, and $v = c^j_{l_1}$, for some clause $c_j = (l_1 \vee l_2 \vee l_3)$. As every literal appears in at least two clauses, we get that there is a clause $c_{j'} \neq c_j$ that contains x_i, in particular, we can write $c_{j'} = (x_i \vee y \vee z)$. Then, as there are no edges connecting different clause-gadgets, we get that the edge $\langle c^{j'}_{x_i}, c^{j'}_{y} \rangle$ is a

[3] For this property, it is sufficient to use the fact that every clause includes literals referring to three different variables.

neighbor edge of u but not a neighbor edge of v. Next, recall that the clause c_j includes literals referring to three different variables, and thus two literals in c_j do not depend on x_i. Assume w.l.o.g that l_2 and l_3 do not depend on x_i. Hence, by the definition of G_θ, there are no edges connecting l_2 and l_3 to x_i and so the edge $\langle c^j_{l_2}, c^j_{l_3}\rangle$ is a neighbor edge of v but not a neighbor edge of u.

Finally, for every variable x_i and clause c_j, there is a literal l that participates in c_j and does not depend on x_i. Also, for every literal l that participates in a clause c_j, there is a variable x_i that l does not depend on, and in both cases it holds that $\neg E(c^j_l, x_i)$. Hence, for every vertex u, there is a vertex v such that $v \neq u$ and $v \in \overline{\eta(u)}$.

A.4 Proof of Lemma 2

We first need some definitions and lemmas. For a vertex $v \in V$ and a state q of $\mathcal{A}_{G,S}$, we say that q is a *v-state* if q is reachable from q_0 upon reading a finite nonempty word that leads to v. The first lemma follows immediately from the definitions:

Lemma 3. *The v-states of $\mathcal{A}_{G,S}$ are $\langle v, 0\rangle$ and $\langle v, 1\rangle$.*

Lemma 4. *Consider a finite word z of the form $v \cdot (\overline{\eta(v)})^*$, for some vertex $v \in V$. The following hold in $\mathcal{A}_{G,S}$.*

1. *The run from q_0 on z reaches the v-state $\langle v, 0\rangle$ and does not traverse α.*
2. *Consider a u-state q_u, for some vertex $u \in \eta(v)$. The finite run r from q_u on z reaches a v-state. In addition, if $v \in S$, then run r traverses α.*

Proof. 1. By definition, $\delta(q_0, v) = \langle v, 0\rangle$. Then, as $\langle v, 0\rangle$ is an $\overline{\alpha}$-state that has a self-loop labeled $\overline{\eta(v)}$, the run of $\mathcal{A}_{G,S}$ on z is of the form $q_0, \langle v, 0\rangle, \ldots, \langle v, 0\rangle$, and thus it reaches $\langle v, 0\rangle$ and does not traverse α.

2. By Lemma 3, we have that $q_u \in \{\langle u, 0\rangle, \langle u, 1\rangle\}$. We distinguish between two cases. First, if $v \in S$, then when $\mathcal{A}_{G,S}$ reads v from q_u, it moves to the state $\delta(q_u, v) = \langle v, 1\rangle$, which is a an α-state. Otherwise, if $v \notin S$, then when $\mathcal{A}_{G,S}$ reads v from q_u, it moves to the state $\delta(q_u, v) = \langle v, 0\rangle$. Then, as reading $\overline{\eta(v)}$ from $\delta(q_u, v)$ leads to the state $\langle v, 0\rangle$ which has a self-loop labeled with $\overline{\eta(v)}$, we get that the run r of q_u on z is of the form $q_u, \delta(q_u, v), \langle v, 0\rangle \ldots, \langle v, 0\rangle$. Hence, if $v \in S$, then r traverses α, and we are done.

□

We can now prove Lemma 2. Thus, we prove that $L(\mathcal{A}_{G,S}) \subseteq L_G$, and $L_G \subseteq L(\mathcal{A}_{G,S})$ iff S is a vertex-cover of G.

We first show that every word in $\overline{L_G}$ is rejected by $\mathcal{A}_{G,S}$. Consider a word w of the form $v_0 \cdot (\overline{\eta(v_0)})^* \cdot v_1 \cdot (\overline{\eta(v_1)})^* \cdot v_2 \cdot (\overline{\eta(v_2)})^* \cdots v_k \cdot (\overline{\eta(v_k)})^\omega$, where for all $0 \leq i \leq k-1$, it holds that $v_{i+1} \in \eta(v_i)$. Let $w = x \cdot y$, where $x = v_0 \cdot (\overline{\eta(v_0)})^* \cdot v_1 \cdot (\overline{\eta(v_1)})^* \cdot v_2 \cdot (\overline{\eta(v_2)})^* \cdots v_k$ and $y = (\overline{\eta(v_k)})^\omega$. An iterative application of Lemma 4 implies that the run of $\mathcal{A}_{G,S}$ on x reaches a v_k-state q_{v_k}. Then, by Lemma 3, it holds that $q_{v_k} \in \{\langle v_k, 0\rangle, \langle v_k, 1\rangle\}$. In both cases, when $\mathcal{A}_{G,S}$ reads y from q_{v_k}, it eventually gets stuck at the $\overline{\alpha}$-state $\langle v_k, 0\rangle$. Hence, the run of $\mathcal{A}_{G,S}$ on $w = x \cdot y$ is rejecting.

Next, note that if $L_G \subseteq L(\mathcal{A}_{G,S})$, then $\mathcal{A}_{G,S}$ recognizes L_G. Then, conisder an edge $\langle u, v\rangle \in E$, and note that the word $w = (u \cdot v)^\omega \in L_G = L(\mathcal{A}_{G,S})$. An iterative

application of Lemma 4 implies that the run of $\mathcal{A}_{G,S}$ on w alternates between u-states and v-states, and as it is accepting, there is u-state or a v-state in $\alpha = S \times \{1\}$. As the latter holds for every edge $\langle u, v\rangle$, it follows that S is a vertex-cover of G. Conversely, assume that S is a vertex-cover of G and consider a word w of the form $w = v_0 \cdot (\overline{\eta(v_0)})^* \cdot v_1 \cdot (\overline{\eta(v_1)})^* \cdot v_2 \cdot (\overline{\eta(v_2)})^* \cdots$, where for all $i \geq 0$, it holds that $v_{i+1} \in \eta(v_i)$. We need to show that $\mathcal{A}_{G,S}$ accepts w. Consider partitioning w into infinitely many infixes $\{z_k\}_{k=0}^{\infty}$, where for all $k \geq 0$, it holds that z_k is of the form $v_k \cdot (\overline{\eta(v_k)})^*$, and $w = z_0 \cdot z_1 \cdot z_2 \cdots$. An iterative application of Lemma 4 implies that the run r of $\mathcal{A}_{G,S}$ on $w = z_0 \cdot z_1 \cdot z_2 \cdots$ moves to a v_k-state upon reading the infix z_k. Also, as the run r traverses α upon reading an infix ($\neq z_0$) corresponding to a state v_k in the vertex-cover S, it holds that r traverses α infinitely often. Indeed, we enter a vertex-cover after reading at most two consecutive infixes.

A.5 Proof of Proposition 1

First, we assume w.l.o.g that $\mathcal{A}$ is SD. Indeed, if $\mathcal{A}$ is not already SD, then we can make it SD without increasing its state-space as in Proposition 3.

We start with Item 1: Consider finite nonempty words x_1 and x_2 that lead to v. We need to show that all the states in $\delta(q_0, x_1) \cup \delta(q_0, x_2)$ are equivalent. As $\mathcal{A}$ is SD, an iterative application of Proposition 2 implies that all the states in $\delta(q_0, x_1)$ are equivalent, and all the states in $\delta(q_0, x_2)$ are equivalent. Also, for all $y \in V^\omega$, we have that $x_1 \cdot y \in L_G$ iff $x_2 \cdot y \in L_G$ iff y is of the form $y = (\overline{\eta(v_0)})^* \cdot v_1 \cdot (\overline{\eta(v_1)})^* \cdot v_2 \cdot (\overline{\eta(v_2)})^* \cdots$, where $v_0 = v$, and for all $i \geq 0$, it holds that $v_{i+1} \in \eta(v_i)$. Hence, all the states in $\delta(q_0, x_1) \cup \delta(q_0, x_2)$ are equivalent.

For Item 2: As G is nice, it is separable, in particular, there is an edge $\langle u_1, u_2\rangle$ that is a neighbor of u but not a neighbor of v. Assume w.l.o.g that $E(u, u_1)$. On one hand, as $u \cdot (u_1 \cdot u_2)^+$ leads to u_2, and $u \cdot (u_1 \cdot u_2)^+ \cdot u_1$ leads to u_1, it follows that $u \cdot (u_1 \cdot u_2)^\omega \in L_G$. On the other hand, as both $v \cdot (u_1 \cdot u_2)^+$ and $v \cdot (u_1 \cdot u_2)^+ \cdot u_1$ lead to v, it follows that $v \cdot (u_1 \cdot u_2)^\omega \notin L_G$. Therefore, the word $z_u = (u_1 \cdot u_2)^\omega$ is such that $u \cdot z_u \in L(\mathcal{A})$ and $v \cdot z_u \notin L(\mathcal{A})$. Hence, z_u is in the language of a u-state in $\delta(q_0, u)$ but not in the language of the v-states in $\delta(q_0, v)$.

For Item 3, assume by way of contradiction that q_0 is a v-state, for some vertex $v \in V$. As G is nice, there is a vertex u, different from v, such that $u \in \overline{\eta(v)}$. On one hand, as q_0 is a v-state, then q_0 is reachable upon reading a finite nonempty word x that leads to v. Then, as $u \in \overline{\eta(v)}$, $x \cdot u$ also leads to v, and thus all the states in $\delta(q_0, u)$ are also v-states. On the other hand, as q_0 is the initial state, it follows that all the states in $\delta(q_0, u)$ are u-states, and we have reached a contradiction to Claim 2 above.

Now, for Item 4, for every vertex $v \in V$, consider the word $w = v^\omega$. As $w \notin L(\mathcal{A})$, then all the runs r of $\mathcal{A}$ on w are such that $inf(r) \cap \alpha = \emptyset$. Now as the word w consists only of v's, every prefix of w leads to v, and thus all the states in $inf(r)$ are v-states. Hence, as $inf(r) \cap \alpha = \emptyset$ and $inf(r) \neq \emptyset$, it follows that all the states in $inf(r)$ are v-states not in α. The above consideration applies to all vertices $v \in V$. Thus, one can define a function f from V to $\{q \in Q \setminus \alpha : q \text{ is a } v\text{-state for some vertex } v \in V\}$ that maps every vertex v to a state in $inf(r)$. As $f(v)$ is a v-state, then, by Claim 2 above, the function f is an injection, and we are done.

Finally, for Item 5, consider the word $w = (u \cdot v)^\omega \in L(\mathcal{A})$, and let r be an accepting run of $\mathcal{A}$ on w. As r is accepting, it holds that $inf(r) \cap \alpha \neq \emptyset$. Since every word in

$(u \cdot v)^* \cdot u$ leads to u, and every word in $(u \cdot v)^+$ leads to v, then the run r moves to a u-state upon reading u and moves to a v-state upon reading v. Thus, $inf(r)$ can contain only u- or v-states, implying there is a v-state or a u-state in α.

A.6 Proof of Theorem 2

We describe a reduction from the vertex-cover problem for nice graphs, proven to be NP-hard in Theorem 1. Given a nice graph $G = \langle V, E \rangle$ and an integer $k \geq 1$, the reduction returns the DBW $\mathcal{A}_{G,V}$ and the integer $|V| + k + 1$. Note that the DBW $\mathcal{A}_{G,V}$ corresponds to the trivial vertex-cover $S = V$; in particular, it recognizes L_G. Also, as DBWs are a special case of GFG-NBWs, then we can view $\mathcal{A}_{G,V}$ as a GFG-NBW. We first prove that the reduction is valid for the minimization of GFG-NBWs. Specifically, we prove that G has a vertex-cover of size at most k iff $\mathcal{A}_{G,V}$ has an equivalent GFG-NBW $\mathcal{B}$ of size at most $|V| + k + 1$. Then, we argue that the proof applies also for the minimization of DBWs.

For the first direction, if $S \subseteq V$ is a vertex-cover of G of size at most k, then, by Lemma 2, the DBW $\mathcal{A}_{G,S}$, which can also be viewed as a GFG-NBW, is of size at most $|V| + k + 1$, and it recognizes L_G. For the other direction, assume that $\mathcal{B}$ is a GFG-NBW of size at most $|V| + k + 1$ that recognizes L_G. By Items 3 and 4 in Proposition 1, it holds that $\mathcal{B}$ has at most k states in $\alpha_{\mathcal{B}}$. To conclude the proof, we show that there is a vertex-cover C of G of size at most $|\alpha_{\mathcal{B}}|$. Consider the set $C = \{v \in V : \mathcal{B} \text{ has a } v\text{-state in } \alpha_{\mathcal{B}} \}$. By Item 5 of Proposition 1, for every edge $\langle u, v \rangle \in E$, at least one of u and v is in C, and hence C is a vertex-cover of G. Finally, note that $|C| \leq |\alpha_{\mathcal{B}}|$. Indeed, consider a function f that maps a vertex $v \in C$ to some v-state in $\alpha_{\mathcal{B}}$. Then, Item 2 of Proposition 1 implies that f is an injection, and thus the reduction is a valid reduction for the minimization of GFG-NBWs.

To conclude that the reduction is also valid for the minimization problem of DBWs. Note that in the first direction of the above proof, the automaton $\mathcal{A}_{G,S}$ is a DBW. Also, a DBW $\mathcal{B}$ of size at most $|V| + k + 1$ that recognizes L_G, is a GFG-NBW, and so the second direction applies also to the DBW case.

A.7 Minimizing GFG-NCWs for DOOM1 languages

We start by extending the *v-state* definition to GFG-NCWs. Consider a nice graph $G = \langle V, E \rangle$, and let $\mathcal{A} = \langle V, Q, q_0, \delta, \alpha \rangle$ be an GFG-NCW that recognizes the language $\overline{L_G}$. For a vertex $v \in V$ and a state $q \in Q$, we say that q is a *v-state* if q is reachable from the initial state q_0 upon reading a finite nonempty word that leads to v. In particular, all the v-successors of q_0 are v-states.

Proposition 4. *For every GFG-NCW $\mathcal{A} = \langle V, Q, q_0, \delta, \alpha \rangle$ that recognizes the language $\overline{L_G}$, the following hold.*

1. *For every vertex $v \in V$, all the v-states are equivalent.*
2. *For every two vertices $u \neq v \in V$, the u-states are not equivalent to the v-states.*
3. *The initial state q_0 is not a v-state, for all vertices $v \in V$.*
4. *$|\{q \in Q \setminus \alpha : q \text{ is a } v\text{-state for some vertex } v \in V\}| \geq |V|$, that is, the number of v-states not in α is at least $|V|$.*
5. *For every edge $\langle u, v \rangle \in E$, there is a v-state or a u-state in α.*

Proof. As we can assume that $\mathcal{A}$ is SD, and the definition of a v-state is the same regardless of the class of the automaton, the proof of Proposition 1 applies, up to a dualization, also here. For example, consider Item 4. For every vertex $v \in V$, consider the word $w = v^\omega$. As $w \in L(\mathcal{A})$, then there is a run r of $\mathcal{A}$ over w such that $inf(r) \cap \alpha = \emptyset$, and the proof proceeds as in Item 4 of Proposition 1. □

We can now prove that the minimization problem is NP-hard already for GFG-NCWs that recognize Doom1 languages. We describe a reduction from the vertex-cover problem for nice graphs, proven to be NP-hard in Theorem 1. Given a nice graph $G = \langle V, E \rangle$ and an integer $k \geq 1$, the reduction returns the DCW $\widetilde{\mathcal{A}}_{G,V}$ and the integer $|V| + k + 1$. Thus, the reduction is identical to that from Theorem 2, with the exception that we view $\mathcal{A}_{G,V}$ as a DCW. Note that the DBW $\mathcal{A}_{G,V}$ corresponds to the trivial vertex-cover $S = V$; in particular, the DCW $\widetilde{\mathcal{A}}_{G,V}$ recognizes $\overline{L_G}$. Also, as DCWs are a special case of GFG-NCWs, then we can view $\widetilde{\mathcal{A}}_{G,V}$ as a GFG-NCW. We prove that G has a vertex-cover of size at most k iff $\widetilde{\mathcal{A}}_{G,V}$ has an equivalent GFG-NCW $\mathcal{B}$ of size at most $|V| + k + 1$.

Correctness of the reduction is similar to that in Theorem 2. Specifically, if $S \subseteq V$ is a vertex-cover of G of size at most k, then, by Lemma 2, the DCW $\widetilde{\mathcal{A}}_{G,S}$, which can be viewed as a GFG-NCW, is of size at most $|V| + k + 1$, and it recognizes $\overline{L_G}$. For the other direction, assume that $\mathcal{B}$ is a GFG-NCW whose size at most $|V| + k + 1$ that recognizes $\overline{L_G}$. Then, we can use Proposition 4 instead of Proposition 1 to show that G have vertex-cover C of size at most k, similarly to proof of Theorem 2.

A.8 Proof of the construction in Theorem 4

First, property 2 follows immediately from the definitions. Then, it is not hard to see that $\mathcal{A}^\#$ is deterministic. Indeed, transitions from the state $q^0_\#$ are deterministic, and by reading # from any state, we move deterministically to q_0.

To conclude the proof, we show that $\mathcal{A}^\#$ recognizes $L(\mathcal{A})^\#$. We first show that $L(\mathcal{A}^\#) \subseteq L(\mathcal{A})^\#$. Let r be an accepting run of $\mathcal{A}^\#$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots$. In particular, r does not get stuck at $q^0_\#$. If r traverses new transitions infinitely often, then w has infinitely many #'s. Otherwise, r eventually leaves the state $q^0_\#$ and moves to q_0 upon reading #. Let $i \geq 0$ be the maximal index such that $\sigma_i = \#$. Since by reading # from any state of $\mathcal{A}^\#$ we move to q_0, it follows that $r_i = q_0$. In particular, $r[i, \infty]$ is a run of $(\mathcal{A}^\#)^{q_0}$ on the suffix $w[i+1, \infty]$. As all the new transitions in $(\mathcal{A}^\#)^{q_0}$ are #-labeled, and q_0 has no incoming transitions in $\mathcal{A}$, it follows that the only way to move to q_0 in $\mathcal{A}^\#$ is upon reading #. Hence, as σ_i is the last # in w, the run $r[i, \infty]$ visits q_0 only once, and hence it is also an accepting run of $\mathcal{A}$ on $w[i+1, \infty]$. Thus, w has a suffix of the form $\# \cdot L(\mathcal{A})$.

For the other direction, let w be a word of the form $(\Sigma \cup \{\#\})^* \cdot \# \cdot L(\mathcal{A}) + \infty\#$. If w has infinitely many #'s, then the run of $\mathcal{A}^\#$ over w is accepting because upon reading a # in any state of $\mathcal{A}^\#$, we visit the state q_0 in $\alpha_\#$. Otherwise, if $w = x \cdot \# \cdot y$, where $y \in L(\mathcal{A})$, then after reading the last # in w, $\mathcal{A}^\#$ moves to q_0 and follows the accepting run of $\mathcal{A}$ over y.

A.9 Example of the construction in Theorem 4

Consider the DBW $\mathcal{A}_{G,S}$ from Fig. 3. Note that the initial state q_0 of $\mathcal{A}_{G,S}$ has no incoming transitions. The DBW $\mathcal{A}^\#_{G,S}$ appears in Fig. 6. As $\mathcal{A}_{G,S}$ recognizes a Live1

language, then $|\mathcal{A}^{\#}_{G,S}| = |\mathcal{A}_{G,S}| + 1$. In addition, note that the only way to reach q_0 in $\mathcal{A}^{\#}_{G,S}$ is by reading $\#$. In particular, reading a word with a suffix of the form $\# \cdot x$, for $x \in \Sigma^\omega$, the DBW $\mathcal{A}^{\#}_{G,S}$ visits q_0 after reading the $\#$, and then its run does not visit q_0, and coincides with the run of $\mathcal{A}_{G,S}$ on x.

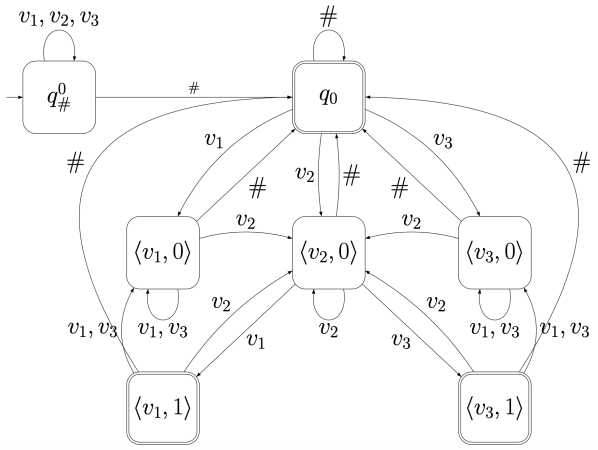

Fig. 6: The DBW $\mathcal{A}^{\#}_{G,S}$.

A.10 Proof of the construction in Theorem 5

We first claim that $\mathcal{A}$ is GFG and recognizes L. First, we show that $L(\mathcal{A}) \subseteq L$. Since $\mathcal{A}$ is embodied in $\mathcal{A}^{\#}$ and $q_\#$ is its initial state, it follows that $L(\mathcal{A}) \subseteq L((\mathcal{A}^{\#})^{q_\#})$. Hence, as $q_\#$ is a $\#$-successor of the initial state of $\mathcal{A}^{\#}$, it follows that $\# \cdot w \in L(\mathcal{A}^{\#})$ for all $w \in L(\mathcal{A})$. Hence, as w is a word over Σ and $L(\mathcal{A}^{\#}) = L^{\#}$, we get that $w \in L$.

Next, we define a strategy $g : \Sigma^* \to Q$ such that for all $w \in L$, it holds that $g(w)$ is an accepting run of $\mathcal{A}$ on w. For all $x \in \Sigma^*$, we define $g(x) = f(\# \cdot x)$. Note that $g(\epsilon) = f(\#) = q_\#$. Also, as f is a GFG strategy for $\mathcal{A}^{\#}$, then for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $\langle f(\# \cdot u), \sigma, f(\# \cdot u \cdot \sigma)\rangle$ is a transition in $\mathcal{A}^{\#}$. Being labeled with a letter from Σ, it is a transition also in $\mathcal{A}$, and thus g is well-defined. Let w be a word in L. Then, as $L(\mathcal{A}^{\#}) = L^{\#}$, we have that $\# \cdot w \in L(\mathcal{A}^{\#})$ and thus the run $f(\# \cdot w) = f(\epsilon), f(\# \cdot w[1,0]), f(\# \cdot w[1,1]), f(\# \cdot w[1,2]), ...$ is an accepting run of $\mathcal{A}^{\#}$. By the definition of g, it holds that $f(\# \cdot w) = f(\epsilon), g(w[1,0]), g(w[1,1]), g(w[1,2]), ...$ and thus $g(w)$ must be accepting as well.

To conclude the proof, assume that L is Live1. Then, we show that we can remove a state from $\mathcal{A}$ and get an equivalent automaton, resulting in an automaton with at most $|\mathcal{A}^{\#}| - 1$ states.

Recall that $\mathcal{A}$ is GFG, and assume that we turn it to an SD automaton as in the proof of Proposition 3. We distinguish between two cases. If we have added a rejecting sink q when semantically determinizing $\mathcal{A}$, then q is unreachable in $\mathcal{A}$ and thus we can remove it. Indeed, if q is reachable upon reading the prefix $x \in \Sigma^*$, then by SDness of $\mathcal{A}$, an iterative application Proposition 2 implies that all the states that are reachable in $\mathcal{A}$ upon reading x are equivalent. Therefore, as $L(\mathcal{A}^q) = \emptyset$, it follows that x is bad

prefix of $L(\mathcal{A}) = L$, contradicting the fact that L is Live1. For the other case, assume that we did not add rejecting sinks when semantically determinizing $\mathcal{A}$. Then, $\mathcal{A}$ is embodied in $\mathcal{A}^\#$. We show that the initial state q_0 of $\mathcal{A}^\#$ is unreachable in $\mathcal{A}$, and so removing it, if it is not already removed, results in an equivalent automaton whose size is at most $|\mathcal{A}^\#| - 1$. Assume by contradiction that q_0 is reachable in $\mathcal{A}$ upon reading a word $x \in \Sigma^*$. On one hand, as $L(\mathcal{A}) = L$ is Live1, there is a $y \in \Sigma^\omega$ such that $x \cdot y \in L(\mathcal{A})$. As $\mathcal{A}$ is SD, an iterative application of Proposition 2 implies that $y \in L(\mathcal{A}^{q_0})$. On the other hand, as $L(\mathcal{A}^\#) = L^\#$ and so $\Sigma^\omega \cap L(\mathcal{A}^\#) = \emptyset$, then all the runs of $\mathcal{A}^\#$ on y are rejecting. Hence, as $\mathcal{A}$ is embodied in $\mathcal{A}^\#$, then all then the runs of of $\mathcal{A}^{q_0}$ on y are also rejecting, contradicting the fact that $y \in L(\mathcal{A}^{q_0})$.

A.11 Proof of Theorem 6

We describe a reduction from the vertex-cover problem for nice graphs, proven to be NP-hard in Theorem 1. Given a nice graph $G = \langle V, E\rangle$ and an integer $k \geq 1$, the reduction returns the DBW $\mathcal{A}^\#_{G,V}$ (that is, the DBW obtained by applying the construction from Theorem 4 on the DBW $\mathcal{A}_{G,V}$ defined in Section 3.1), and the integer $|V|+k+2$. Note that as DBWs are a special case of GFG-NBWs, we can view $\mathcal{A}^\#_{G,V}$ also as a GFG-NBW. Note also that the initial state of $\mathcal{A}_{G,V}$ has no incoming transitions, as required. We first prove that the reduction is valid for the minimization of GFG-NBWs. Specifically, we prove that G has a vertex-cover of size at most k iff $\mathcal{A}^\#_{G,V}$ has an equivalent GFG-NBW of size at most $|V| + k + 2$. Then, we argue that the same considerations apply for the minimization of DBWs.

For the first direction, if $S \subseteq V$ is a vertex-cover of size at most k of G. Then, by Lemma 2, the DBW $\mathcal{A}_{G,S}$, whose size is at most $|V|+k+1$, recognizes L_G. Hence, by Theorem 4, the DBW $\mathcal{A}^\#_{G,S}$, which can be viewed also as a GFG-NBW, is of size is at most $|V| + k + 2$, and it recognizes $L(\mathcal{A}_{G,S})^\# = L^\#_G$. For the other direction, assume that $\mathcal{A}^\#$ is a GFG-NBW of size at most $|V| + k + 2$ that recognizes $L^\#_G$. As L_G is Live1, we get by Theorem 5 that there is a GFG-NBW $\mathcal{A}$ of size at most $|V|+k+1$ that recognizes L_G. By the correctness of the reduction from Section 3.1, it follows that G has a vertex-cover of size at most k, and we are done. As the construction of $\mathcal{A}$ preserves determinizm, the proof applies also to the DBW case.

A.12 Proof of Theorem 7

Let $\mathcal{A}^\# = \langle \Sigma \cup \{\#\}, Q, q_0, \delta, \alpha\rangle$, and let $f : (\Sigma \cup \{\#\})^* \to Q$ be a strategy witnessing the GFGness of $\mathcal{A}^\#$. As in the proof of Theorem 5, we obtain $\mathcal{A}$ from $\mathcal{A}^\#$ by removing all #-transitions and choosing $q_\# = f(\#)$ to be its initial state. Clearly, $\mathcal{A}$ is over Σ and determinization of $\mathcal{A}^\#$ is preserved in $\mathcal{A}$.

Next, consider an infinite word $w \in \Sigma^\omega$. As $L(\mathcal{A}^\#) = \overline{L^\#} = (\Sigma \cup \{\#\})^* \cdot \# \cdot \overline{L} + \Sigma^\omega$, then $\# \cdot w \in L(\mathcal{A}^\#)$ iff $w \in \overline{L}$. Therefore, one can prove that $\mathcal{A}$ is GFG and recognizes $\overline{L}$ as in Theorem 5, up to a dualization: specifically, replacing L and $L^\#$ there with $\overline{L}$ and $\overline{L^\#}$, respectively, results in a valid argument.

To conclude the proof, assume that $\overline{L}$ is *Doom*1. Then, we show that the inital state q_0 of $\mathcal{A}^\#$ is unreachable in $\mathcal{A}$ and thus it can be removed from $\mathcal{A}$ and we get an equivalent automaton whose size is at most $|\mathcal{A}^\#| - 1$. Assume by contradiction that q_0 is reachable in $\mathcal{A}$ upon reading a word $x \in \Sigma^*$. On one hand, as $L(\mathcal{A}) = \overline{L}$ is

DOOM1, there is $y \in \Sigma^\omega$ such that $x \cdot y \notin L(\mathcal{A})$. Hence, it follows that $y \notin L(\mathcal{A}^{q_0})$. On the other hand, as $L(\mathcal{A}^\#) = \overline{L^\#}$ and so $\Sigma^\omega \subseteq L(\mathcal{A}^\#)$, then there is an accepting r run of $\mathcal{A}^\#$ over y. As y has no #'s then r exists also in $\mathcal{A}$, contradicting the fact that $y \notin L(\mathcal{A}^{q_0})$.

A.13 Proof of Theorem 8

As stated earlier, hardness of minimizing DCWs that recognize DOOM2 languages follows by duallity of the Büchi and co-Büchi acceptance conditions. Next, we consider minimizing GFG-NCWs that recognize DOOM2 languages. We describe a reduction from the vertex-cover problem for nice graphs, proven to be NP-hard in Theorem 1. Given a nice graph $G = \langle V, E\rangle$ and an integer $k \geq 1$, the reduction returns the DCW $\widetilde{\mathcal{A}}^\#_{G,V}$ and the integer $|V| + k + 2$. Thus, the reduction is identical to that from Theorem 6, with the exception that we view $\mathcal{A}^\#_{G,V}$ as a DCW. Note that as DCWs are a special case of GFG-NCWs, we can view $\widetilde{\mathcal{A}}^\#_{G,V}$ as a GFG-NCW. We prove that G has a vertex-cover of size at most k iff $\widetilde{\mathcal{A}}^\#_{G,V}$ has an equivalent GFG-NCW of size at most $|V| + k + 2$.

For the first direction, if $S \subseteq V$ is a vertex-cover of size at most k of G. Then, by Lemma 2, the DBW $\mathcal{A}_{G,S}$, whose size is at most $|V|+k+1$, recognizes L_G. Hence, by Theorem 4, the DCW $\widetilde{\mathcal{A}}^\#_{G,S}$, which can be viewed also as a GFG-NCW, is of size is at most $|V| + k + 2$, and it recognizes $\overline{L(\mathcal{A}_{G,S})^\#} = \overline{L^\#_G}$. For the other direction, assume that $\mathcal{A}^\#$ is a GFG-NCW of size at most $|V| + k + 2$ that recognizes $\overline{L^\#_G}$. As L_G is LIVE1, then $\overline{L_G}$ is DOOM1. Hence, we get by Theorem 7 that there is a GFG-NCW $\mathcal{A}$ of size at most $|V| + k + 1$ that recognizes $\overline{L_G}$. By the correctness of the reduction from Theorem 3, it follows that G has a vertex-cover of size at most k, and we are done.

A.14 Proof of Theorem 9

Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$. We adapt the construction from Theorem 4 to GFG-tNBWs. Specifically, if $L(\mathcal{A})$ is LIVE1, then $\mathcal{A}^\# = \langle \Sigma \cup \{\#\}, Q \cup \{q^0_\#\}, q^0_\#, \delta_\#, \alpha_\#\rangle$ is obtained from $\mathcal{A}$ by adding a new state $q^0_\#$ that has a Σ-labeled self-loop that goes with # to the state q_0, to which we move upon reading # from all states. In addition, $\alpha_\# = \alpha \cup ((Q \cup \{q^0_\#\}) \times \{\#\} \times \{q_0\})$. Note that unlike the construction in Theorem 4, as the acceptance is transition-based, it is okay for $\mathcal{A}$ to have transitions entering its initial state, as we prove below.

Also here, if $L(\mathcal{A})$ is not LIVE1, then $\mathcal{A}$ must contain a state q with $L(\mathcal{A}^q) = \emptyset$, and w.l.o.g we assume that q is a rejecting sink in $\mathcal{A}$. We then define $\mathcal{A}^\#$ as above, except that instead of defining the state $q^0_\#$ as a new state, we define $q^0_\# = q$. In particular, in either case, the initial state $q^0_\#$ has a Σ-labeled self-loop that goes with # to q_0.

We prove that $\mathcal{A}^\#$ satisfies the three properties. First, property 3 follows immediately from the definitions. Then, it is not hard to see that $\mathcal{A}^\#$ preserves determinism. Indeed, transitions from the state $q^0_\#$ are deterministic, and by reading # from any state, we move deterministically to q_0. Thus, property 2 holds.

To conclude the proof, we show that $\mathcal{A}^{\#}$ is a GFG-tNBW that recognizes $L(\mathcal{A})^{\#}$. We first show that $L(\mathcal{A}^{\#}) \subseteq L(\mathcal{A})^{\#}$. Let r be an accepting run of $\mathcal{A}^{\#}$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots$. If r traverses new transitions infinitely often, then w has infinitely many #'s. Otherwise, r eventually leaves the state $q^0_{\#}$ and moves to q_0 upon reading #. Let $i \geq 0$ be the maximal index such that $\sigma_i = \#$. Since by reading # from any state of $\mathcal{A}^{\#}$ we move to q_0, it follows that $r_i = q_0$. In particular, $r[i, \infty]$ is a run of $(\mathcal{A}^{\#})^{q_0}$ on the suffix $w[i+1, \infty]$. As all the new transitions in $(\mathcal{A}^{\#})^{q_0}$ are #-labeled, and σ_i is the last # in w, it follows that $r[i, \infty]$ traverses only transitions of $\mathcal{A}$, and thus is an accepting run of $\mathcal{A}$ on $w[i+1, \infty]$. Hence, w has a suffix of the form $\# \cdot L(\mathcal{A})$.

Next, we show that there is a strategy $g : (\Sigma \cup \{\#\})^* \to Q \cup \{q^0_{\#}\}$ such that for all words $w \in L(\mathcal{A})^{\#}$, we have that $g(w)$ is an accepting run of $\mathcal{A}^{\#}$ on w. Let f be a strategy witnessing $\mathcal{A}$'s GFGness. Note that nondeterminism has to be resolved by g only in the states in $Q \setminus \{q^0_{\#}\}$ and words ending by a letter in Σ. Indeed, transitions from the state $q^0_{\#}$ are deterministic, and so are #-transitions. Essentially, the strategy g tries to follow the strategy f after seeing a #. Formally, for all $x \in (\Sigma \cup \{\#\})^*$ and $y \in \Sigma^*$, we define $g(y) = q^0_{\#}$ and $g(x \cdot \# \cdot y) = f(y)$. As transitions from $q^0_{\#}$ are deterministic, and $q^0_{\#}$ has a Σ-labeled self-loop, then the only state $\mathcal{A}^{\#}$ can reach upon reading a word $y \in \Sigma^*$ is $q^0_{\#}$. Also, by reading # in $\mathcal{A}^{\#}$ we move to q_0, in particular, all runs of $\mathcal{A}^{\#}$ on $x \cdot \#$ for some prefix x lead to q_0. Thus, as g follows $f(y)$ upon reading a prefix of the form $x \cdot \# \cdot y$, where $y \in \Sigma^*$, then g is well-defined. Now let w be a word of the form $(\Sigma \cup \{\#\})^* \cdot \# \cdot L(\mathcal{A}) + \infty\#$. If w has infinitely many #'s, then every run of $\mathcal{A}^{\#}$ over w is accepting because upon reading a # in any state of $\mathcal{A}^{\#}$, we traverse $\alpha_{\#}$, in particular $g(w)$ is accepting. Otherwise, if $w = x \cdot \# \cdot y$, where $y \in L(\mathcal{A})$, then after reading the last # in w, $g(w)$ moves to q_0. Also, by the definition of g, it holds that $g(x \cdot \# \cdot y[1,0]), g(x \cdot \# \cdot y[1,1]), g(x \cdot \# \cdot y[1,2]), ... = f(y[1,0]), f(y[1,1]), f(y[1,2]), ...$. Hence, as $f(y)$ is accepting, then $g(x \cdot \# \cdot y[1,0]), g(x \cdot \# \cdot y[1,1]), g(x \cdot \# \cdot y[1,2]), ...$ traverses $\alpha_{\#}$ infinitely often and thus $g(w)$ is accepting as well.

A.15 Proof of Theorem 10

For the first direction, assume that $\mathcal{B}$ is a GFG-tNBW of size at most k that recognizes $L(\mathcal{A})$. We distinguish between two cases. If $L(\mathcal{A})$ is Live1, then by Theorem 9, it holds that $\mathcal{B}^{\#}$ is a GFG-tNBW equivalent to $\mathcal{A}^{\#}$ whose size is at most $|\mathcal{B}| + 1$. As $|\mathcal{B}| + 1 \leq k + 1 = k'$, we are done. For the other case, if $L(\mathcal{A})$ is not Live1, then also $L(\mathcal{B})$ is not Live1. Then, by Theorem 9, it holds that $\mathcal{B}^{\#}$ is a GFG-tNBW equivalent to $\mathcal{A}^{\#}$ whose size is at most $|\mathcal{B}|$. As $|\mathcal{B}| \leq k = k'$, we are done.

For the other direction, assume that $\mathcal{B}^{\#}$ is a GFG-tNBW equivalent to $\mathcal{A}^{\#}$ and is of size at most k'. Theorem 9 implies that $L(\mathcal{B}^{\#}) = L(\mathcal{A})^{\#}$. We distinguish between two cases. If $L(\mathcal{A})$ is not Live1, then $k' = k$ and by Theorem 5 (which applies also to GFG-tNBWs) there is a GFG-tNBW $\mathcal{B}$ that recognizes $L(\mathcal{A})$ and is of size at most $|\mathcal{B}^{\#}| \leq k' = k$. For the other case, if $L(\mathcal{A})$ is Live1, then $k' = k + 1$. In this case, by Theorem 5 we can assume further that the GFG-tNBW $\mathcal{B}$ is of size at most $|\mathcal{B}^{\#}| - 1 \leq k' - 1 = k$.

Now, since the constructions in Theorems 9 and 5 both preserve determinism, and tDBWs are a special case of GFG-tNBWs, the result applies also to tDBWs.

A.16 Proof of Theorem 11

We claim that the same reduction from Theorem 10 applies also here. Specifically, given a tDCW $\mathcal{A}$ and an integer $k \geq 1$, the reduction returns the tDCW $\widetilde{(\widetilde{\mathcal{A}})^{\#}}$ and the integer k', where $k' = k$ if $\overline{L(\mathcal{A})}$ is not LIVE1, and $k' = k+1$ if $\overline{L(\mathcal{A})}$ is LIVE1. We prove that $\mathcal{A}$ has an equivalent tDCW of size at most k iff $\widetilde{(\widetilde{\mathcal{A}})^{\#}}$ has an equivalent tDCW of size at most k'. The proof follows by the correctness of the reduction from Theorem 10 and the duality of the Büchi and co-Büchi acceptance conditions. Both directions are similar, and we show only one:

Let $\mathcal{B}$ be a tDCW whose size is at most k that is equivalent to $\mathcal{A}$, then by duality, $\widetilde{\mathcal{B}}$ is a tDBW whose size is at most k that is equivalent to $\widetilde{\mathcal{A}}$. As $\widetilde{\mathcal{A}}$ recognizes $\overline{L(\mathcal{A})}$, we get by the definition of k' and the correctness of the reduction from Theorem 10, that $(\widetilde{\mathcal{A}})^{\#}$ has an equivalent tDBW $\mathcal{C}$ whose size is at most k'. Hence, by duality, we get that $\widetilde{\mathcal{C}}$ is a tDCW whose size is at most k' that recongnizes $\overline{L((\widetilde{\mathcal{A}})^{\#})}$. In particular, $\widetilde{\mathcal{C}}$ is equivalent to $\widetilde{(\widetilde{\mathcal{A}})^{\#}}$ and we are done.

A.17 Proof of the claim in Theorem 12

Let $R_x = a^+ \cdot x \cdot (x+y)^*$, $R_y = a^+ \cdot y \cdot (x+y)^*$, and $R = (R_x \cdot R_x + R_y \cdot R_y)$. We prove that $\mathcal{A}$ recognizes the LIVE3 language ∞R, is GFG, yet not DBP.

Note that the only nondeterminsitic choice in $\mathcal{A}$ is when the letter a is read in the state q_0. By removing the α-transitions of $\mathcal{A}$, we get two components, namely, $S_1 = \{q_0, q_1, q_2\}$ and $S_2 = Q \setminus S_1$. It is not hard to see that a run r in $\mathcal{A}$ is accepting iff r does not get stuck in a component.

We prove that $\mathcal{A}$ is a GFG-tNBW that recognizes ∞R. To begin with, we show that $L(\mathcal{A}) \subseteq \infty R$. Consider an accepting run r of $\mathcal{A}$ on an infinite word w. As r is accepting, it does not get stuck in a component. In particular, r has infinitely many infixes $r[i,j]$ such that $r_i = r_j = q_0$, for all $i < k < j$, $r_k \neq q_0$, and $r[i,j]$ enters the component S_2. We show that $r[i,j]$ is a finite run on a word that has an infix in R. The only way $r[i,j]$ enters S_2 is by moving to q_3 or to q_4. In addition, $r[i,j]$ leaves S_2 by traversing a path of the form $q_3 \to q_5^* \to q_0$, reading an infix in R_x, or a path of the form $q_4 \to q_6^* \to q_0$, reading an infix in R_y. Therefore, as we can reach q_3 only by reading an infix in R_x, and we can reach q_4 only by reading an infix in R_y, it follows that $r[i,j]$ runs on a word having an infix in R.

Next, note that the above considerations imply that by reading an infix in R in S_2, we move to S_1. Also, by reading an infix in R from q_0, we get stuck in S_1 only by choosing the same successor of q_0 twice. Therefore, as every suffix of a word in ∞R is in ∞R, it follows that the following strategy, in $\mathcal{A}$'s single nondeterministic state q_0, results in accepting all words in ∞R: "if the last read chunk of letters are in R_x then go to q_1 else go to q_2".

Now, to see that $\mathcal{A}$ is not DBP, recall that the only nondeterminism of $\mathcal{A}$ is in q_0. Therefore, there are two possible prunings to consider: the DBW $\mathcal{A}'$ in which $\delta(q_0, a) = q_1$ and the DBW $\mathcal{A}''$ in which $\delta(q_0, a) = q_2$. With the former, any word in $(R_y)^\omega$ is in $L(\mathcal{A}) \setminus L(\mathcal{A}')$ and with the latter any word in $(R_x)^\omega$ is in $L(\mathcal{A}) \setminus L(\mathcal{A}'')$.

Finally, we argue that every GFG-tNBW $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, q^0_\mathcal{A}, \Delta_\mathcal{A}, \alpha_\mathcal{A} \rangle$ that is not DBP, can be converted to an equivalent GFG-NBW $\mathcal{B} = \langle \Sigma, Q_\mathcal{A} \times \{0,1\}, \langle q^0_\mathcal{A}, 0 \rangle, \delta_\mathcal{B}, \alpha_\mathcal{B} \rangle$, for $\alpha_\mathcal{B} = Q_\mathcal{A} \times \{1\}$, which is not DBP. The GFG-NBW $\mathcal{B}$ is obtained from $\mathcal{A}$ by taking

two copies of the state space, one of them is accepting, and defining the transition, for every state $q \in Q_{\mathcal{A}}$ and letter $\sigma \in \Sigma$, as follows.

$$\delta_{\mathcal{B}}(\langle q, 0\rangle, \sigma) = \delta_{\mathcal{B}}(\langle q, 1\rangle, \sigma) = \{\langle s, 0\rangle : \langle q, \sigma, s\rangle \in \Delta_{\mathcal{A}} \setminus \alpha_{\mathcal{A}}\} \cup \{\langle s, 1\rangle : \langle q, \sigma, s\rangle \in \alpha_{\mathcal{A}}\}.$$

It is easy to see that by projecting a run of $\mathcal{B}$ on $Q_{\mathcal{A}}$, we get a run of $\mathcal{A}$. In addition, runs $r_{\mathcal{A}} = r_0, r_1, \ldots$ over $w = \sigma_1 \cdot \sigma_2 \cdots$ in $\mathcal{A}$, correspond to runs $r_{\mathcal{B}} = \langle r_0, j_0\rangle, \langle r_1, j_1\rangle, \ldots$ in $\mathcal{B}$, where $\langle r_i, \sigma_{i+1}, r_{i+1}\rangle \in \alpha_{\mathcal{A}}$ iff $j_{i+1} \in \alpha_{\mathcal{B}}$. Hence, the construction preserves not only GFGness of $\mathcal{A}$ but also strategies witnessing GFGness in $\mathcal{A}$. In particular $\mathcal{A}$ is DBP iff $\mathcal{B}$ is DBP.

A.18 Proof of Theorem 13

We prove the correctness of the construction, thus $L(\mathcal{B}) = L(\mathcal{A})$, and $\mathcal{B}$ is a minimal GFG-NCW equivalent to $\mathcal{A}$.

First, we assume that all the states of $\mathcal{A}$ are reachable, and that $\mathcal{A}$ is SD. Indeed, detecting reachable states is easy, and by Proposition 3, semantically determinizing $\mathcal{A}$ can be done in polynomial time. Thus, we do not lose generality by the latter assumptions. Then, note that the latter assumptions imply that all the states of $\mathcal{A}$ are equivalent. To see why, let q be a state of $\mathcal{A}$. We show that $L(\mathcal{A}^q) = L(\mathcal{A})$. Let $x \in \Sigma^*$ be such that $q \in \delta(q_0, x)$, and let $y \in L(\mathcal{A}^q)$. As $L(\mathcal{A}) = \neg\infty R$, it follows that $x \cdot y \in \neg\infty R$; and hence $y \in \neg\infty R$. That is, $y \in L(\mathcal{A})$ and thus $L(\mathcal{A}^q) \subseteq L(\mathcal{A})$. For the other inclusion, let y be a word in $L(\mathcal{A})$. Then, $y \in \neg\infty R$. Hence, $x \cdot y \in \neg\infty R$ and thus $x \cdot y \in L(\mathcal{A})$. As $\mathcal{A}$ is SD, an iterative application of Proposition 2 implies that $y \in L(\mathcal{A}^q)$.

Now, correctness and minimality follow from Propositions 5 and 6 below.

Proposition 5. *$\mathcal{B}$ is a GFG-NCW equivalent to $\mathcal{A}$.*

Proof. If $L(\mathcal{A})$ is trivial, then $\mathcal{B}$ is a one state DCW equivalent to $\mathcal{A}$; in particular, being deterministic, $\mathcal{B}$ is GFG. We proceed with the case where $L(\mathcal{A})$ is non-trivial. We first show that $L(\mathcal{B}) \subseteq L(\mathcal{A})$. Let $r = r_0, r_1, \ldots$ be an accepting run of $\mathcal{B}$ on a word w. Then, there is $i \geq 0$ such that $r[i, \infty]$ is a run on the suffix $w[i+1, \infty]$ that does not visit s. By the definition of $\mathcal{B}$, the run $r[i, \infty]$ is a run of $\mathcal{U}^{r_i}$ that does not traverse $\alpha_{\mathcal{U}}$-transitions in $\mathcal{U}$. In particular, $w[i+1, \infty] \in L(\mathcal{U}^{r_i}) \subseteq L(\mathcal{U})$ where the last containment follows from the fact that all the states of $\mathcal{U}$ are equivalent. Note that indeed, this property of tNCWs for Doom3 languages is maintained by the minimization algorithm of [1].

Next, we show that $L(\mathcal{A}) \subseteq L(\mathcal{B})$ and that $\mathcal{B}$ is GFG by defining a strategy $g : \Sigma^* \to Q_{\mathcal{U}} \cup \{s\}$ such that for every word $w \in L(\mathcal{U})$, it holds that $g(w)$ is an accepting run of $\mathcal{B}$ on w. Let $f : \Sigma^* \to Q_{\mathcal{U}}$ be a strategy witnessing $\mathcal{U}$'s GFGness. We define now the strategy g. As $\mathcal{U}$ is safe-deterministic, we get that the only nondeterminisim in $\mathcal{B}$, is upon reading a letter σ from the state s. The strategy g essentially tries to follow f when possible. Formally, if $g(x) = s$ for some $x \in \Sigma^*$, then for all $\sigma \in \Sigma$, we define $g(x \cdot \sigma) = f(x \cdot \sigma)$. Otherwise, if $g(x) \neq s$, then for all $\sigma \in \Sigma$, $g(x \cdot \sigma)$ is defined as follows: if there are no $\alpha_{\mathcal{U}}$ σ-labeled transitions going out from $g(x)$ in $\mathcal{U}$, then $g(x \cdot \sigma)$ equals the state in $\delta_{\mathcal{U}}(g(x), \sigma)$, otherwise, $g(x \cdot \sigma) = s$. Clearly, the strategy g is well-defined as we can non-deterministically move to any state of $\mathcal{U}$ upon reading any letter from s. Also, as $\mathcal{U}$ is safe-deterministic, it follows that if there are no $\alpha_{\mathcal{U}}$ σ-labeled

transitions going out from a state $q \neq s$, then $\delta_{\mathcal{U}}(q, \sigma)$ is a singleton. Now consider a word $w \in L(\mathcal{U})$ and the accepting run $f(w)$ of $\mathcal{U}$ on w. As $f(w)$ is accepting, there is $i \geq 0$ such that $f(w[1,i]), f(w[1,i+1]), f(w[1,i+2]), \ldots$ is a run that does not traverse $\alpha_{\mathcal{U}}$ on the suffix $w[i+1,\infty]$. Next, we show that $g(w[1,i]), g(w[1,i+1]), g(w[1,i+2]), \ldots$ visits s at most once, and thus $g(w)$ is accepting. If $g(w[1,j]) = s$ for some $j \geq i$, then by the definition of g, it holds that $g(w[1,j+1]) = f(w[1,j+1])$. Hence, as $f(w[1,i]), f(w[1,i+1]), f(w[1,i+2]), \ldots$ does not traverse $\alpha_{\mathcal{U}}$, $\mathcal{U}$ is safe-deterministic and safe-homogenous, then it follows that $f(w[1,j+1]), f(w[1,j+2]), f(w[1,j+3]), \ldots$ is the run that $g(w)$ eventually follows, and thus $g(w)$ eventually never visits s. □

Proposition 6. *$\mathcal{B}$ is a minimal GFG-NCW.*

Proof. If $L(\mathcal{A})$ is trivial, then $\mathcal{B}$ is a one state DCW, and thus is a minimal GFG-NCW. We proceed with the case where $L(\mathcal{A})$ is non-trivial. Recall that $|\mathcal{B}| = |\mathcal{U}| + 1$. In order to show that $\mathcal{B}$ is minimal, we show that every minimal GFG-NCW $\mathcal{C}$ that recognizes $L(\mathcal{A})$ is of size at least $|\mathcal{U}| + 1$.

Consider a minimal GFG-NCW $\mathcal{C}$ for $L(\mathcal{A})$. As $\mathcal{C}$ can be viewed as a GFG-tNCW and $\mathcal{U}$ is a minimal GFG-tNCW, it follows that $|C| \geq |\mathcal{U}|$. Assume by contradiction that $|C| = |\mathcal{U}|$, and let $\alpha_{\mathcal{C}}$ be the acceptance condition of $\mathcal{C}$, thus the set of states that an accepting run should visit only finitely often.

Consider the GFG-tNCW $\mathcal{D}$, with acceptance condition $\alpha_{\mathcal{D}}$, that is obtained from $\mathcal{C}$ by defining all the transitions that leave an $\alpha_{\mathcal{C}}$-state as $\alpha_{\mathcal{D}}$-transitions. As $|\mathcal{D}| = |\mathcal{C}| = |\mathcal{U}|$, then $\mathcal{D}$ is a minimal GFG-tNCW. In particular, all states of $\mathcal{D}$ are reachable. Also, we assume that $\mathcal{D}$ is SD (otherwise, we turn it into SD without increasing its state-space, as explained in Proposition 3). The following properties hold in $\mathcal{D}$.

1. As $L(\mathcal{D})$ is non-trivial, it must be the case that $\alpha_{\mathcal{D}} \neq \emptyset$.
2. Consider a state q in $\mathcal{D}$. If there is an $\alpha_{\mathcal{D}}$-transition going out from q, then all the runs of $\mathcal{D}^q$ traverse $\alpha_{\mathcal{D}}$ transitions. [4]
3. As $\mathcal{D}$ recognizes a Doom3 language, is SD and all its states are reachable, then all its states are equivalent.

We show next that there exist equivalent states p and q in $\mathcal{D}$ such that p belongs to a nonempty cycle that does not traverse $\alpha_{\mathcal{D}}$-transitions, and all runs of $\mathcal{D}^q$ traverse $\alpha_{\mathcal{D}}$ transitions. First, Property 3 above implies that every two states p and q are equivalent. Then, the existence of q is immediate from Properties 1 and 2 above. Finally, the fact that there is a state p of $\mathcal{D}$ that belongs to a nonempty cycle that does not traverse $\alpha_{\mathcal{D}}$-transitions is immediate from the fact that $L(\mathcal{D})$ is non-trivial. Indeed, if all the cycles that pass through p traverse α_D, for all states p of $\mathcal{D}$, then $L(\mathcal{D})$ would have been empty.

The minimization algorithm of [1] is such that applying it on a GFG-tNCW that has two states p and q as above, it removes the state q and results in a strictly smaller GFG-tNCW. In particular, applying the minimization algorithm of [1] on $\mathcal{D}$ results in a strictly smaller automaton, contradicting the minimality of $\mathcal{D}$. □

[4] We note that turning $\mathcal{D}$ into SD does interfere with this property - if we add a rejecting sink s when turning $\mathcal{D}$ into SD, then we can require that all transitions that lead to s are also in $\alpha_{\mathcal{D}}$.