Minimising Good-For-Games Automata Is NP-Complete

Sven Schewe
University of Liverpool, UK
https://cgi.csc.liv.ac.uk/~sven/
sven.schewe@liverpool.ac.uk

Abstract

This paper discusses the hardness of finding minimal good-for-games (GFG) Büchi, Co-Büchi, and parity automata with state based acceptance. The problem appears to sit between finding small deterministic and finding small nondeterministic automata, where minimality is NP-complete and PSPACE-complete, respectively. However, recent work of Radi and Kupferman has shown that minimising Co-Büchi automata with transition based acceptance is tractable, which suggests that the complexity of minimising GFG automata might be *cheaper* than minimising deterministic automata.

We show for the standard state based acceptance that the minimality of a GFG automaton is NP-complete for Büchi, Co-Büchi, and parity GFG automata. The proofs are a surprisingly straight forward generalisation of the proofs from deterministic Büchi automata: they use a similar reductions, and the same hard class of languages.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Good-for-Games Automata, Automata Minimisation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.56

Related Version An earlier version is available on arXiv [9], https://arxiv.org/abs/2003.11979.

Funding This work was partly supported by the EPSRC through grant EP/P020909/1.

Acknowledgements Many thanks to Patrick Totzke and Karoliina Lehtinen for valuable feedback and pointers to beautiful related works, as well as the constructive feedback of the reviewers.

1 Introduction

Good-for-games (GFG) automata form a useful class of automata that can be used to replace deterministic automata to recognise languages in several settings, like reactive synthesis [4]. As good-for-games automata sit between deterministic and general nondeterministic automata, it stands to be expected that the complexity of their minimality also sits between the minimality of deterministic automata (NP-complete [8]) and nondeterministic automata (which is PSPACE-complete like for nondeterministic finite automata [5]). It thus came as a surprise when Radi and Kupferman showed that minimising Co-Büchi automata with transition based acceptance is tractable [7].

This raises the question whether the difference is that good-for-games automata are inherently simpler to minimise, or if it is a consequence of choosing the less common transition based acceptance. We show that the answer for the more common state based acceptance is that minimising GFG automata is as hard as minimising deterministic automata.

While extending our "inclusion in NP" argument to transition based acceptance is straight forward, our hardness proof generalises the NP-hardness proof from [8], and we will close this paper with discussing why this hardness argument does not extend to automata with transition based acceptance. This leaves the complexity of minimising transition based GFG automata (except for Co-Büchi GFG automata [7]) and deterministic automata open.

40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020).
Editors: Nitin Saxena and Sunil Simon; Article No. 56; pp. 56:1–56:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Automata

2.1 Nondeterministic Parity Automata

Parity automata are word automata that recognise ω-regular languages over a finite set of symbols. A *nondeterministic parity automaton* (NPA) is a tuple $\mathcal{P} = (\Sigma, Q, q_0, \delta, \pi)$, where

- Σ denotes a finite set of symbols,
- Q denotes a finite set of states,
- $q_0 \in Q_+$ with $Q_+ = Q \cup \{\bot, \top\}$ denotes a designated initial state,
- $\delta : Q_+ \times \Sigma \to 2^Q_+$ (with $2^Q_+ = 2^Q \cup \{\{\bot\}, \{\top\}\} \setminus \{\emptyset\}$ is a function that maps pairs of states and input letters to either a non-empty set of states, or to $\bot$ (false, immediate rejection, blocking) or $\top$ (true, immediate acceptance)[1], such that $\delta(\top, \sigma) = \{\top\}$ and $\delta(\bot, \sigma) = \{\bot\}$ hold for all $\sigma \in \Sigma$, and
- $\pi : Q_+ \to P \subset \mathbb{N}$ is a priority function that maps states to natural numbers (mapping $\bot$ and $\top$ to an odd and even number, respectively), called their *priority*.

Parity automata read infinite **input words** $\alpha = a_0 a_1 a_2 \ldots \in \Sigma^\omega$. (As usual, $\omega = \mathbb{N}_0$ denotes the non-negative integers.) Their acceptance mechanism is defined in terms of runs: a run $\rho = r_0 r_1 r_2 \ldots \in {Q_+}^\omega$ of $\mathcal{P}$ on α is an ω-word that satisfies $r_0 = q_0$ and, for all $i \in \omega$, $r_{i+1} \in \delta(r_i, a_i)$. A run is called *accepting* if the highest number occurring infinitely often in the infinite sequence $\pi(r_0)\pi(r_1)\pi(r_2)\ldots$ is even, and *rejecting* if it is odd. An ω-word is *accepted* by $\mathcal{P}$ if it has an accepting run. The set of ω-words accepted by $\mathcal{P}$ is called its *language*, denoted $\mathcal{L}(\mathcal{P})$. Two automata that recognise the same language are called *language equivalent*.

We assume without loss of generality that $\max\{P\} \leq |Q| + 1$. (If a priority $p \geq 2$ does not exist, we can reduce the priority of all states whose priority is strictly greater than p by 2 without affecting acceptance.)

2.2 Büchi and Co-Büchi Automata

Büchi and Co-Büchi automata – abbreviated NBAs and NCAs – are NPAs where the image of the priority function π is contained in $\{1, 2\}$ and $\{2, 3\}$, respectively. In both cases, the automaton is often denoted $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, where $F \subseteq Q_+$ is called (the set of) *final states* and denotes those states with the higher priority (2 for Büchi, 3 for Co-Büchi). The remaining states $Q_+ \setminus F$ are called *non-final* states.

2.3 Deterministic and Good-for-Games Automata

An automaton is called *deterministic* if the image of the transition function δ consists only of singletons (i.e. is included in $\{\{q\} \mid q \in Q_+\}$. For convenience, δ is therefore often viewed as a function $\bar{\delta} : Q_+ \times \Sigma \to Q_+$ (with $\delta(q, \sigma) \mapsto \{\bar{\delta}(q, \sigma)\}$).

A nondeterministic automaton is called *good-for-games* (GFG) if it only relies on a limited form of nondeterminism: GFG automata can make their decision of how to resolve their nondeterministic choices on the *history* at any point of a run – rather than using the knowledge of the complete word as a nondeterministic automaton normally would – without changing their language. They can be characterised in many ways, including as automata that simulate deterministic automata.

[1] The question whether or not an automaton can immediately accept or reject is a matter of taste. Often, immediate rejection is covered by allowing δ to be partial while there is no immediate acceptance. We allow both – so $\top$ and $\bot$ are not counted as states – but treat them as accepting and rejecting sink states, respectively, for technical convenience.

We use the following formalisation: a nondeterministic automaton $\mathcal{P} = (\Sigma, Q, q_0, \delta, \pi)$ is good-for-games if there is function $\nu : q_0 {Q_+}^* \Sigma \to Q_+$ such that, for every infinite word $\alpha = a_0 a_1 a_2 \ldots \in \Sigma^\omega$, $\mathcal{P}$ has an accepting run ρ' if, and only if, it has an accepting run $\rho = r_0 r_1 r_2 \ldots \in {Q_+}^\omega$ with $r_0 = q_0$ and, for all $i \in \mathbb{N}_0$, $r_{i+1} = \nu(r_0, \ldots, r_i; a_i)$.

Broadly speaking, a good-for-games automaton sits in the middle between a nondeterministic and a deterministic automaton: $\mathcal{P}$ and ν together define a deterministic automaton (if such a ν exists, there is a finite state one), but as the ν does not have to be explicitly provided, $\mathcal{P}$ can be more succinct than a deterministic automaton.

2.4 Automata Transformations & Conventions

For an NPA $\mathcal{B} = (\Sigma, Q, q_0, \delta, \pi)$ and a state $q \in Q_+$, we denote with $\mathcal{B}_q = (\Sigma, Q, q, \delta, \pi)$ the automaton resulting from $\mathcal{B}$ by changing the initial state to q.

There are two standard measures for the size of an automaton $\mathcal{P} = (\Sigma, Q, q_0, \delta, \pi)$: the number $|Q|$ of its states, and the size $\sum\limits_{q \in Q, a \in \Sigma} |\delta(q, a)|$ of its transition table.

3 Main Result

We show the following theorem.

▶ **Theorem 1.** *The following problems are NP-complete (all 20 combinations).*

1. *Given a good-for-games / deterministic parity / Büchi / Co-Büchi automaton and a bound k, is there a language equivalent good-for-games parity automaton with at most k states / entries in its transition table?*
2. *Given a good-for-games / deterministic Büchi automaton and a bound k, is there a language equivalent good-for-games Büchi automaton with at most k states / entries in its transition table?*
3. *Given a good-for-games / deterministic Co-Büchi automaton and a bound k, is there a language equivalent good-for-games Co-Büchi automaton with at most k states / entries in its transition table?*

The 20 individual questions are, of course, all very similar. Note, however, that in the ten cases where a good-for-games automaton is given, its good-for-games property is not checked; instead we simply do not require the result to be correct where the given automaton is not good-for games. In particular, the complexity of determining GFG-ness remains an open research question (except for Büchi [1] and Co-Büchi [6] automata, where it is known to be tractable).

For inclusion in NP (Section 4), the small good-for-games automaton can be guessed, and the guess can be validated with standard simulation games (Corollary 5).

NP hardness is established in Section 5 (Theorem 15), it turns out that the known hardness proof for deterministic Büchi and Co-Büchi automata can be adjusted to good-for-games automata, providing hardness for all combinations of our main theorem.

4 Inclusion in NP

We start with re-visiting a standard simulation game between a verifier, who wants to establish language inclusion through simulation, and a spoiler, who wants to destroy the proof. Note that the spoiler does not try to disprove language inclusion, but merely wants to show that it cannot be established through simulation.

4.1 Simulation Game

For two NPAs $\mathcal{P}^1 = (\Sigma, Q_1, q_0^1, \delta_1, \pi_1)$ and $\mathcal{P}^2 = (\Sigma, Q_2, q_0^2, \delta_2, \pi_2)$, we define the "$\mathcal{P}^2$ simulates $\mathcal{P}^1$" game, where a spoiler intuitively tries to show that $\mathcal{P}^1$ accepts a word not in the language of $\mathcal{P}^2$, as follows.

The game is played on $Q_1 \times Q_2 \cup Q_1 \times \Sigma \times Q_2$ and starts in (q_0^1, q_0^2). In a state $(q_1, q_2) \in Q_1 \times Q_2$, the spoiler selects a letter $\sigma \in \Sigma$ and a σ successor $q_1' \in \delta(q_1, \sigma)$ of q_1 for $\mathcal{P}^1$ and moves to (q_1', σ, q_2). In a state $(q_1', \sigma, q_2) \in Q_1 \times \Sigma \times Q_2$, the verifier selects a σ successor $q_2' \in \delta(q_2, \sigma)$ of q_2 for $\mathcal{P}^2$ and moves to (q_1', q_2').

Verifier and spoiler will together produce a play $(q_0^1, q_0^2)(q_1^1, a_0, q_0^2)(q_1^1, q_1^2)(q_2^1, a_1, q_1^2)$ $(q_2^1, q_2^2)(q_3^1, a_2, q_2^2)(q_3^1, q_3^2)(q_4^1, a_3, q_3^2)\ldots$. The verifier wins if, and only if, the run $q_0^1 q_1^1 q_2^1 q_3^1 \ldots$ of $\mathcal{P}^1$ is rejecting *or* the run $q_0^2 q_1^2 q_2^2 q_3^2 \ldots$ of $\mathcal{P}^2$ is accepting.

Simulation games have been used to validate GFG-ness right from their introduction [4].

▶ **Lemma 2.** *If the* verifier *wins the* $\mathcal{P}^2$ simulates $\mathcal{P}^1$ *game, then she wins positionally, and checking if she wins is in NP.*

This is a standard inclusion game, and similar games have e.g. been used in [6].

Proof. The verifier plays a game with two disjunctive (from verifier's perspective) parity conditions (as the complement of a parity condition is a parity condition). A parity condition is in particular a Rabin condition, and the disjunction of Rabin conditions is still a Rabin condition. Thus, if the verifier can meet her parity objective, she can do so positionally[2] [3]. Thus, it suffices to guess the winning strategy of the verifier, and then check (in P)[3] if the spoiler wins his resulting one player game with two conjunctive parity conditions. ◀

▶ **Lemma 3.** *Given an NPA* $\mathcal{P}^1$ *and a good-for-games NPA* $\mathcal{P}^2$*, checking* $\mathcal{L}(\mathcal{P}^1) \subseteq \mathcal{L}(\mathcal{P}^2)$ *is in NP.*

Proof. Consider the "$\mathcal{P}^2$ simulates $\mathcal{P}^1$" game played on an NPA $\mathcal{P}^1 = (\Sigma, Q_1, q_0^1, \delta_1, \pi_1)$ and a good-for-games NPA $\mathcal{P}^2 = (\Sigma, Q_2, q_0^2, \delta_2, \pi_2)$.

We first show that spoiler wins this if there is a word $\alpha \in \mathcal{L}(\mathcal{P}^1) \setminus \mathcal{L}(\mathcal{P}^2)$: in this case, spoiler can guess such a word alongside an accepting run for $\mathcal{P}^1$ for α – note that there is no accepting run of $\mathcal{P}^2$ for α, as $\alpha \notin \mathcal{L}(\mathcal{P}^2)$.

We finally show that verifier wins this game if $\mathcal{L}(\mathcal{P}^2) \supseteq \mathcal{L}(\mathcal{P}^1)$. In this case, verifier can construct the run $q_0^2 q_1^2 q_2^2 q_3^2 \ldots$ on the word α the spoiler successively produces. Moreover, as $\mathcal{P}^2$ is good-for-games, the verifier can do this independent of the transitions the spoiler selects, basing her choices instead on her good-for-games strategy ν^2. If α is in $\mathcal{L}(\mathcal{P}^2)$, then $q_0^2 q_1^2 q_2^2 q_3^2 \ldots$ is accepting and verifier wins. If α is not in $\mathcal{L}(\mathcal{P}^2)$, then α is not in $\mathcal{L}(\mathcal{P}^1) \subseteq \mathcal{L}(\mathcal{P}^2)$ either; thus $q_0^1 q_1^1 q_2^1 q_3^1 \ldots$ is rejecting and verifier wins. ◀

▶ **Theorem 4.** *Given an NPA* $\mathcal{P}^1$ *and a good-for-games NPA* $\mathcal{P}^2$*, checking if* $\mathcal{P}^1$ *is good-for-games* and *satisfies* $\mathcal{L}(\mathcal{P}^1) = \mathcal{L}(\mathcal{P}^2)$ *is in NP.*

---

[2] A strategy is called positional if it only depends on the current state, not on the history of how one got there.

[3] This problem is actually in NL, as the spoiler can guess the pair of winning priorities and guess a lasso-like path with an initial part, and a repeating part that starts and ends in the same state and has the correct dominating priorities for both parity conditions. (This does not have to be a cycle, as it might be necessary to visit a state once for establishing the dominating priority for either parity condition.)

A similar game is used in [6] to establish that GFG-ness can be decided in EXPTIME. The new observation here is the inclusions in NP, assuming a GFG automaton is provided.

Proof. We first use the previous lemma to check $\mathcal{L}(\mathcal{P}^1) \subseteq \mathcal{L}(\mathcal{P}^2)$ in NP. For the rest of the proof, we assume that this test has been passed, such that $\mathcal{L}(\mathcal{P}^1) \subseteq \mathcal{L}(\mathcal{P}^2)$ was established.

We then play the same game with inverse roles, i.e. the "$\mathcal{P}^1$ simulates $\mathcal{P}^2$" game. The question if verifier wins is again in NP.

If $\mathcal{L}(\mathcal{P}^1) \neq \mathcal{L}(\mathcal{P}^2)$ holds, then the already established $\mathcal{L}(\mathcal{P}^1) \subseteq \mathcal{L}(\mathcal{P}^2)$ entails that there is a word $\alpha \in \mathcal{L}(\mathcal{P}^2) \setminus \mathcal{L}(\mathcal{P}^1)$. In this case spoiler can win by guessing such a word $\alpha \in \mathcal{L}(\mathcal{P}^2) \setminus \mathcal{L}(\mathcal{P}^1)$ alongside an accepting run for $\mathcal{P}^2$ for α – note that there is no accepting run of $\mathcal{P}^1$ for α in this case, regardless of whether or not $\mathcal{P}^1$ is good-for games.

If $\mathcal{P}^1$ is good-for-games *and* $\mathcal{L}(\mathcal{P}^1) = \mathcal{L}(\mathcal{P}^2)$ holds, then verifier wins (because $\mathcal{L}(\mathcal{P}^2) \subseteq \mathcal{L}(\mathcal{P}^1)$ can be verified in NP using Lemma 3).

Finally, if $\mathcal{L}(\mathcal{P}^1) = \mathcal{L}(\mathcal{P}^2)$ holds and verifier wins, then $\mathcal{P}^1$ is good-for-games: this is because a winning strategy – like the positional strategy that exists (Lemma 2) – for verifier in the "$\mathcal{P}^1$ simulates $\mathcal{P}^2$" game transforms a good-for-games strategy ν^2 for $\mathcal{P}^2$ into a good-for-games strategy ν^1 for $\mathcal{P}^1$, and $\mathcal{P}^1$ can simply emulate the behaviour of $\mathcal{P}^2$ using the (positional) winning strategy from of the verifier. ◀

This provides all upper bounds of Theorem 1 when taking into account that k should be smaller than the provided automaton. (If it is not, then the answer is always "yes", as the automaton itself can be used.).

▶ **Corollary 5.** *All problems from Theorem 1 can be solved in NP.*

Note that this does not result in a test whether or not a given automaton is good-for-games, it merely allows, given a good-for-games automaton, to validate that a second NPA is both: good-for-games *and* language equivalent.

For Büchi [1] and Co-Büchi automata [6], it is tractable to check whether or not an automaton is good-for-games.

5 NP Hardness

In this section we generalise the hardness argument for the minimality of deterministic Büchi and Co-Büchi automata from [8]. It lifts the reduction from the problem of finding a minimal vertex cover of a graph to the minimisation of deterministic Büchi automata to a reduction to the minimisation of good-for-games automata. (A vertex cover is a set of vertices that covers at least one end point of every edge.) In the graph from Figure 1, the vertices in red and the vertices in white are both vertex covers, and the red vertices are the only minimal vertex cover. The reduction first defines the characteristic language of a simple connected graph; for technical convenience it assumes a distinguished initial vertex.

We show that the states of a good-for-games Büchi (or parity) automaton that recognises this characteristic language must satisfy side-constraints, which imply that it has at least $2n + k$ states, where n is the number of vertices of the graph, and k is the size of its minimal vertex cover. Moreover, from a good-for-games automaton with s states, we can infer a vertex cover with size at most $s - 2n$.

At the same time, it is simple to construct, for a given vertex cover of size k, a deterministic Büchi automaton of size $2n + k$ that recognises the characteristic language of this graph. (Figure 3 shows such a DBA for the example from Figure 1.) This holds in particular for the trivial vertex cover (which contains all vertices) that results in a DBA with $3n$ states.

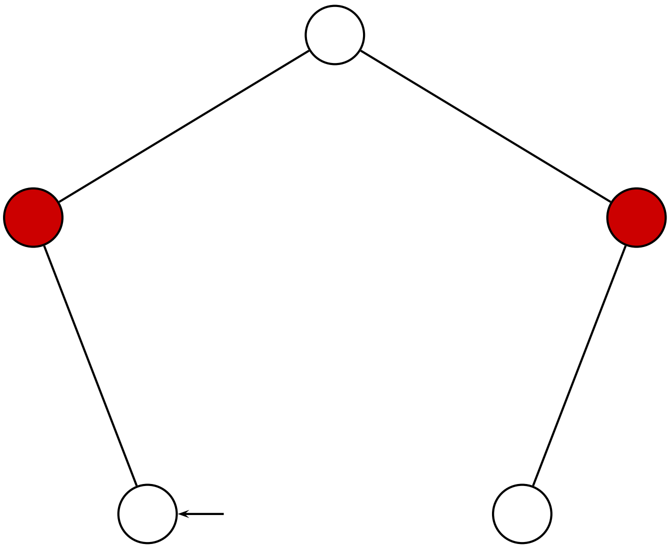

Figure 1 A nice graph (a connected graph with a dedicated initial vertex) with a 2 vertex cover (in red). *Is a nice graph k coverable?* is an NP-complete problem.

Minimising the automaton defined by this trivial vertex cover can therefore be used to determine a minimal vertex cover for a given simple connected graph, which concludes the NP hardness argument.

Finally we show how to adjust the argument for minimal Co-Büchi automata, which – different to deterministic automata, where one can simply use the dual automaton – requires a small adjustment in the definition of the characteristic language for good-for-games automata.

Returning to the reduction known from deterministic automata, we call a non-trivial ($|V| > 1$) simple undirected connected graph $\mathcal{G}_{v_0} = (V, E)$ with a distinguished initial vertex $v_0 \in V$ *nice*. The restriction to nice graphs leaves the problem of finding a minimal vertex cover NP-complete.

▶ **Lemma 6** ([8]). *The problem of checking whether a nice graph $\mathcal{G}_{v_0}$ has a vertex cover of size k is NP-complete.*

Following [8], we define the *characteristic language* $\mathcal{L}(\mathcal{G}_{v_0})$ of a nice graph $\mathcal{G}_{v_0}$ as the ω-language over $V_\natural = V \cup \{\natural\}$ (where $\natural$ indicates a stop of the evaluation in the next step – it can be read "stop") consisting of

1. all ω-words of the form ${v_0}^*{v_1}^+{v_2}^+{v_3}^+{v_4}^+ \ldots \in V^\omega$ with $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, (words where $v_0, v_1, v_2, \ldots$ form an infinite path in $\mathcal{G}_{v_0}$), and
2. all ω-words that start with[4] ${v_0}^*{v_1}^+{v_2}^+ \ldots {v_n}^+\natural v_n \in {V_\natural}^*$ with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$. (Words where $v_0, v_1, v_2, \ldots, v_n$ form a finite – and potentially trivial – path in $\mathcal{G}_{v_0}$, followed by a $\natural$ sign, followed by the last vertex of the path $v_0, v_1, v_2, \ldots, v_n$, and by v_0 if $\natural$ was the first letter.)

We call the ω-words in (1) *trace-words*, and those in (2) *$\natural$-words*. The trace-words are in V^ω, while the $\natural$-words are in ${V_\natural}^\omega \setminus V^\omega$.

Figure 2 shows a deterministic Büchi automaton that recognises the $\natural$-words for the nice graph from Figure 1. The five colours are used as names (or: identifiers) for the vertices

4 this includes words that start with $\natural v_0$

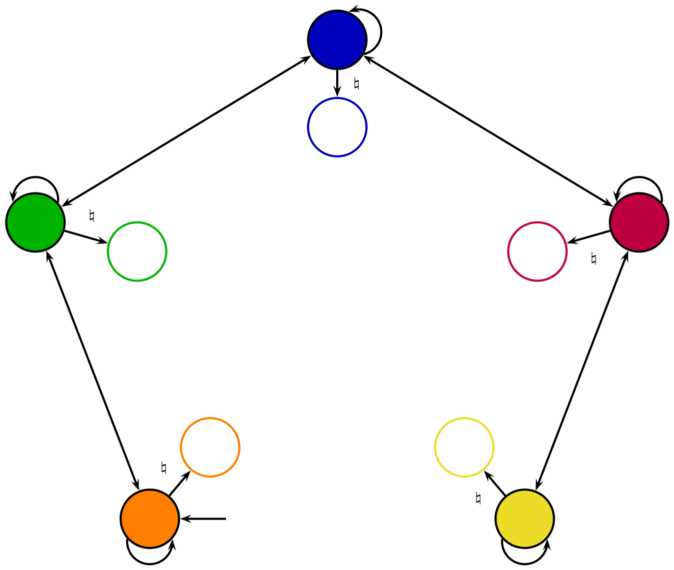

Figure 2 A deterministic Büchi automaton that recognises the ♮-words for the nice graph from Figure 1. The five colours are used as names for the vertices of the nice graph. The colour of the full (outer) vertices intuitively reflects the colour of the *previous* vertex seen while traversing an input word that can still be completed to an accepted ♮ word. If the automaton reads a vertex (identified by its colour), which identifies the current vertex or a vertex adjacent to it, it updates the stored vertex to the one it has read; it blocks (moves to $\bot$) when reading a different vertex.
When reading ♮, it moves to the light inner vertex while keeping the stored colour/vertex, shown by the colour of its fringe. From a light (inner) vertex, it accepts (moves to $\top$) if it sees the stored vertex (indicated by the colour of the fringe) next, and blocks (moves to $\bot$) otherwise.

of the nice graph. The colour of the full (outer) vertices intuitively reflects the colour of the *previous* vertex seen while traversing an input word that can still be completed to an accepted ♮ word (initialised to the colour of the dedicated initial vertex of the nice graph, in this case, •.) If the automaton reads a vertex (here identified by its colour), which identifies either the current vertex or a vertex adjacent to it, it updates the stored vertex to the one it has read. If it reads a different vertex, which is not adjacent, it blocks (moves to $\bot$).

When reading ♮, it moves to a light (inner) vertex while keeping the stored colour of the last vertex seen vertex, shown by the colour of its fringe. From a light (inner) vertex, it accepts (moves to $\top$) if it sees the stored vertex (indicated by the colour of the fringe) next, and blocks (moves to $\bot$) otherwise.

A word that starts with • • • • • • • • • • • • ♮ •, for example, is accepted, while words that start with • • • • • • • • • • • • ♮ • (wrong colour after ♮) or • • • • • • • • • • • • ♮ • (• and • are not adjacent) are rejected.

Let $\mathcal{B}$ be a parity good-for-games automaton that recognises the characteristic language of $\mathcal{G}_{v_0} = (V, E)$. We call a state of $\mathcal{B}$

- a *v-state* if it can be reached upon an input word ${v_0}^*{v_1}^+{v_2}^+\ldots{v_n}^+ \in V^*$, with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, that ends in $v = v_n$ (in particular, the initial state of $\mathcal{B}$ is a v_0-state), and
- a *v♮-state* if it can be reached from a v-state upon reading a ♮ sign.

We call the union over all v-states the set of *vertex-states*, and the union over all v♮-states the set of ♮-states.

It is not hard to define, for a given nice graph $\mathcal{G}_{v_0} = (V, E)$ with vertex cover C, a *deterministic* Büchi automaton $\mathcal{B}_C^{\mathcal{G}_{v_0}} = (V_\natural, (V \times \{n, \natural\}) \cup (C \times \{f\}), (v_0, n), \bar{\delta}, (C \times \{f\}) \cup \{\top\})$ with $2|V| + |C|$ states that recognises the characteristic language of $\mathcal{G}_{v_0}$ [8]. (The n and f in the state refer to non-final and final, respectively.) We simply choose

- $\bar{\delta}\big((v, n), v'\big) = (v', f)$ if $\{v, v'\} \in E$ and $v' \in C$,
  $\bar{\delta}\big((v, n), v'\big) = (v', n)$ if $\{v, v'\} \in E$ and $v' \notin C$,
  $\bar{\delta}\big((v, n), v'\big) = (v, n)$ if $v = v'$,
  $\bar{\delta}\big((v, n), v'\big) = (v, \natural)$ if $v' = \natural$, and
  $\bar{\delta}\big((v, n), v'\big) = \bot$ otherwise;
- $\bar{\delta}\big((v, f), v'\big) = \bar{\delta}\big((v, n), v'\big)$, and
- $\bar{\delta}\big((v, \natural), v\big) = \top$ and $\bar{\delta}\big((v, \natural), v'\big) = \bot$ for $v' \neq v$.

$\mathcal{B}_C^{\mathcal{G}_{v_0}}$ simply has one $v\natural$-state for each vertex $v \in V$ of $\mathcal{G}_{v_0}$, one final v-state for each vertex in the vertex cover C, and one non-final v-state for each vertex $v \in V$ of $\mathcal{G}_{v_0}$. It moves to the final (accepting) copy (v, f) for a vertex $v \in C$ of a v-state only upon taking an edge to v, but not on a repetition of v.

Figure 3 shows a Büchi automaton that recognises the characteristic language of the nice graph from Figure 1. Different from the automaton from Figure 2, it also has to consider the trace-words, who stay in the 7 outer states (depicted as fully coloured in).

The accepting states define a cover, and a cover can be used to select final states – the automaton from Figure 3 moves to a final state whenever it "enters a vertex" from the cover shown in Figure 1. This way, every (after stuttering) infinite path sees infinitely many final states, while every (after stuttering) finite path does not. If the defining set was not a cover, then there were two adjacent states that are both not part of the cover, and the infinite path that goes back and forth between them would not be accepted.

▶ **Lemma 7** ([8]). *For a nice graph $\mathcal{G}_{v_0} = (V, E)$ with initial vertex v_0 and vertex cover C, the Büchi automaton $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ recognises the characteristic language of $\mathcal{G}_{v_0}$.*

Having seen how to get from a cover to an automaton that recognises the characteristic language of a nice graph, we now study the other direction.

▶ **Lemma 8.** *Let $\mathcal{G}_{v_0} = (V, E)$ be a nice graph with initial vertex v_0, and let $\mathcal{B} = (V_\natural, Q, q_0, \delta, \pi)$ be a good-for-games parity automaton that recognises the characteristic language of $\mathcal{G}_{v_0}$. Then the following holds:*

1. *for all v in V, there is a v-state from which all words that start with $\natural v$ are accepted – we call these states the* core v-states*;*
2. *for all v in V, there is a core v-state with an odd priority;*
3. *for all $v \in V$ and $w \in V_\natural$ with $v \neq w$ and for every v-state q_v, words that start with $\natural w$ are not in the language of $\mathcal{B}_{q_v}$;*
4. *for all v in V, there is a $v\natural$-state from which all words that start with v are accepted – we call these states the* core $v\natural$-states*;*
5. *for all v in V and $w \in V_\natural$ with $v \neq w$ and for every $v\natural$-state $q_{v\natural}$, words that start with w are not in the language of $\mathcal{B}_{q_{v\natural}}$; and*
6. *for every edge $\{v, w\} \in E$, there is a v-state or a w-state with an even priority.*

Proof. **1.** Let $v = v_n$ and let $v_0, v_1, v_2, ..., v_n$ be a path in $\mathcal{G}_{v_0}$. As $\mathcal{B}$ recognises $\mathcal{L}(\mathcal{G}_{v_0})$ and is good-for-games, it must, after having read the first $n + 1$ or more letters of an input word $v_0, v_1, v_2, ..., {v_n}^\omega$ (using its good-for-games strategy ν), with $\{v_i, v_{i+1}\} \in E$ for all $i < n$, be in a core v-state, as words that start with this and continue with $\natural v$ are in $\mathcal{L}(\mathcal{G}_{v_0})$.

2. Furthermore, the run $\mathcal{B}$ produces (using ν) for $v_0, v_1, v_2, ..., v_n{}^\omega$ has a dominating priority determined by its tail of core v-states, and the core v-state with the highest priority that occurs infinitely many times must have an odd priority (as the word is not in $\mathcal{L}(\mathcal{G}_{v_0})$). Consequently, there must be at least one core v-state with an odd priority.
3. If (3) does not hold, a witness would provide a word accepted by $\mathcal{B}$ but not in $\mathcal{L}(\mathcal{G}_{v_0})$.
4. Let $v = v_n$ and let $v_0, v_1, v_2, ..., v_n$ be a path in $\mathcal{G}_{v_0}$. As $\mathcal{B}$ recognises $\mathcal{L}(\mathcal{G}_{v_0})$ and is GFG, it must, after having read the first $n+2$ letters of an input word that starts with $v_0, v_1, v_2, ..., v_n, \natural$ (using its good-for-games strategy ν), with $\{v_i, v_{i+1}\} \in E$ for all $i < n$, be in a core $v\natural$-state, as words that start with this and continue with v are in $\mathcal{L}(\mathcal{G}_{v_0})$.
5. If (5) does not hold, a witness would provide a word accepted by $\mathcal{B}$ but not in $\mathcal{L}(\mathcal{G}_{v_0})$.
6. Let us consider an arbitrary edge $\{v, w\} \in E$, $v = v_n$, and the run of $\mathcal{B}$ (following ν) on $v_0, v_1, v_2, \ldots, v_n, (w, v)^\omega$ in $\mathcal{L}(\mathcal{G}_{v_0})$ (i.e. for all $i < n$. $\{v_i, v_{i+1}\} \in E$).
   The run must be accepting, and, as argued in (1), once the word alternates between v and w, the run alternates between core v-states and core w-states. Thus, the core v-state *or* the core w-state with the highest priority that occurs infinitely often must have an even priority. ◀

The sixth claim implies that the set C of vertices with a core vertex-state with even priority is a vertex cover of $\mathcal{G}_{v_0} = (V, E)$. Thus, $\mathcal{B}$ has at least $|C|$ core vertex states with an even priority. (1–3) provide that $\mathcal{B}$ has at least $|V|$ vertex-states with odd priority, and it follows with (4+5) that there are $|V|$ core $\natural$-states that are disjoint from the core vertex-states:

▶ **Corollary 9.** *For a good-for-games parity automaton $\mathcal{B} = (V_\natural, Q, q_0, \delta, \pi)$ with s states that recognises the characteristic language of a nice graph $\mathcal{G}_{v_0} = (V, E)$ with initial vertex v_0, the set $C = \{v \in V \mid$ there is a v-state with an even priority$\}$ is a vertex cover of $\mathcal{G}_{v_0}$, and $\mathcal{B}$ has at least $2|V| + |C|$ states ($s \geq 2|V| + |C|$), such that $|C| \leq s - 2|V|$ holds.* ◀

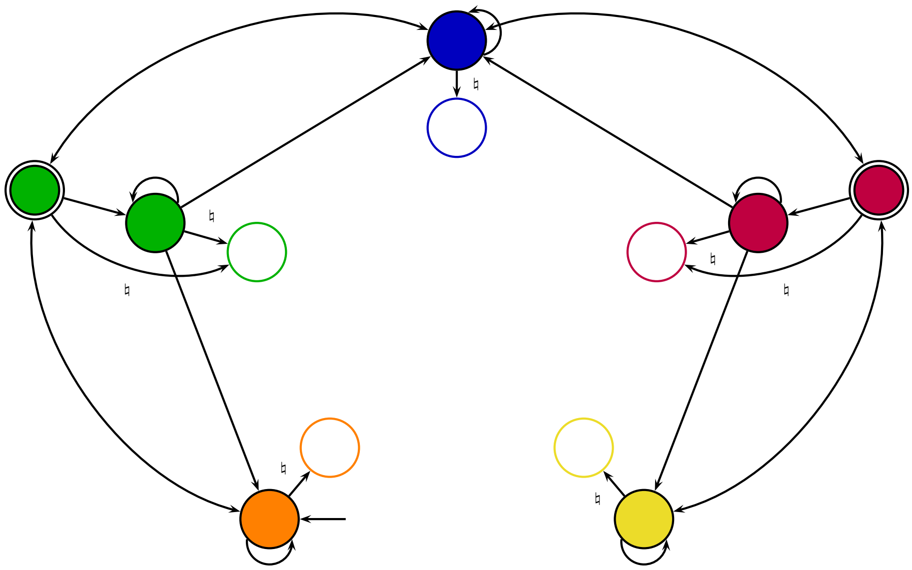

Figure 3 A minimal GFG (and deterministic) Büchi automaton that recognises the characteristic language of the nice graph from Figure 1. For a nice graph $\mathcal{G}_{v_0} = (V, E)$, a GFG parity automaton that recognises its characteristic language needs $|V|$ states reached after reading (the first) $\natural$ (the light, inner states), $|V|$ states with odd priority reachable prior to reading the first $\natural$, and, broadly speaking, sufficiently many states with even priority, such that they identify a cover. The Büchi automaton shown here is defined by the cover that contains the red states shown in Figure 1.

Corollary 9 and Lemma 7 immediately imply:

▶ **Corollary 10.** *Let C be a minimal vertex cover of a nice graph $\mathcal{G}_{v_0} = (V, E)$. Then $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ is a minimal deterministic Büchi automaton that recognises the characteristic language of $\mathcal{G}_{v_0}$, and there is no good-for-games parity automaton with less states than $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ that recognises the same language. Moreover, every minimal good-for-games automaton identifies a cover C' with $|C'| = |C|$.* ◀

This suffices for most cases from Theorem 1, but not for the cases where the automaton given is a Co-Büchi automaton. To also cover Co-Büchi automata, we change the characteristic language to the *adjusted language* $\mathcal{L}'(\mathcal{G}_{v_0})$ of a nice graph $\mathcal{G}_{v_0}$ as the ω-language over $V_\natural = V \cup \{\natural\}$ that consists of

1. all ω-words of the form ${v_0}^*{v_1}^+{v_2}^+{v_3}^+{v_4}^+ \ldots {v_n}^\omega \in V^\omega$ with $\{v_i, v_{i+1}\} \in E$ for all $i < n$, (words where $v_0, v_1, v_2, \ldots, v_n$ form a finite (possibly trivial) path in $\mathcal{G}_{v_0}$), and
2. all ω-words that start with[5] ${v_0}^*{v_1}^+{v_2}^+ \ldots {v_n}^+\natural v_n \in {V_\natural}^*$ with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$. (Words where $v_0, v_1, v_2, \ldots, v_n$ form a finite – and potentially trivial – path in $\mathcal{G}_{v_0}$, followed by a $\natural$ sign, followed by the last vertex of the path $v_0, v_1, v_2, \ldots, v_n$, and by v_0 if $\natural$ was the first letter.)

▶ **Lemma 11.** *Let $\mathcal{G}_{v_0} = (V, E)$ be a nice graph with initial vertex v_0, and let $\mathcal{B} = (V_\natural, Q, q_0, \delta, \pi)$ be a good-for-games parity automaton that recognises the adjusted language $\mathcal{L}'(\mathcal{G}_{v_0})$ of $\mathcal{G}_{v_0}$. Then the following holds:*

1. *for all v in V, there is a v-state from which all words that start with $\natural v$ are accepted – we call these states the* core v-states*;*
2. *for all v in V, there is a core v-state with an* ***even*** *priority;*
3. *for all $v \in V$ and $w \in V_\natural$ with $v \neq w$ and for every v-state q_v, words that start with $\natural w$ are not in the language of $\mathcal{B}_{q_v}$;*
4. *for all v in V, there is a $v\natural$-state from which all words that start with v are accepted – we call these states the* core $v\natural$-states*;*
5. *for all v in V and $w \in V_\natural$ with $v \neq w$ and for every $v\natural$-state $q_{v\natural}$, words that start with w are not in the language of $\mathcal{B}_{q_{v\natural}}$; and*
6. *for every edge $\{v, w\} \in E$, there is a v-state or a w-state with an* ***odd*** *priority.*

The changes in the proof compared to Lemma 8 are simply to replace even and odd accordingly.

With the same argument as before we get the same corollary:

▶ **Corollary 12.** *For a good-for-games parity automaton with s states that recognises the adjusted characteristic language of a nice graph $\mathcal{G}_{v_0} = (V, E)$ with initial vertex v_0, the set $C = \{v \in V \mid \text{there is a } v\text{-state with an even priority}\}$ is a vertex cover of $\mathcal{G}_{v_0}$, and $\mathcal{B}$ has at least $2|V| + |C|$ states ($s \geq 2|V| + |C|$), such that $|C| \leq s - 2|V|$ holds.*

▶ **Lemma 13.** *For a nice graph $\mathcal{G}_{v_0} = (V, E)$ with initial vertex v_0 and vertex cover C, the Co-Büchi automaton[6] $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ recognises the adjusted language of $\mathcal{G}_{v_0}$.*

Proof. We argue separately that the trace-words and $\natural$-words accepted by $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ are exactly the trace-words and $\natural$-words, respectively, in $\mathcal{L}'(\mathcal{G}_{v_0})$.

---

[5] this includes words that start with $\natural v_0$

[6] The automaton is the same as before, but read as a Co-Büchi automaton.

For a *trace-word* $\alpha = v_1 v_2 v_3 \ldots \in V^\omega$, $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ has the run $(v_0, n)(v_1, x_1)(v_2, x_2)(v_3, x_3)\ldots$ (with $x_i \in \{n, f\}$ for all $i \in \mathbb{N}$) if, for all $i \in \mathbb{N}$, either $v_{i-1} = v_i$ or $\{v_{i-1}, v_i\} \in E$ holds; otherwise the automaton blocks (has a tail of $\bot$ states) at $i_{\min}$-th letter, where $i_{\min}$ is the minimal i such that $v_{i-1} \neq v_i$ and $\{v_{i-1}, v_i\} \notin E$. A trace-word where the automaton blocks is rejected by $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ and not in $\mathcal{L}'(\mathcal{G}_{v_0})$.

We now consider those trace-words, for which $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ does not block. For these words, we call the set $I = \{i \in \mathbb{N} \mid \{v_{i-1}, v_i\} \in E\}$ transition indices. Now $\alpha \in \mathcal{L}'(\mathcal{G}_{v_0})$ holds if, and only if, I is finite. If I is finite, we call its maximal element $i_{\max}$, and set $i_{\max}$ to 0 if I is empty. The run of $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ on α is then $(v_0, n)(v_1, x_1)\ldots(v_{i_{\max}-1}, x_{i_{\max}-1})(v_{i_{\max}}, x_{i_{\max}})(v_{i_{\max}}, n)^\omega$; it has a tail of non-final states $(v_{i_{\max}}, n)$, and α is therefore accepted by $\mathcal{B}_C^{\mathcal{G}_{v_0}}$.

If I is infinite, we use the infinite ascending chain $i_1 < i_2 < i_3 < \ldots$ with $I = \{i_n \mid n \in \mathbb{N}\}$. Then, for all $k \in \mathbb{N}$, $v_{i_k - 1} \neq v_{i_k} = v_{i_{k+1}-1} \neq v_{i_{k+1}}$ holds and $\{v_{i_k}, v_{i_{k+1}}\} \in E$. $\{v_{i_k}, v_{i_{k+1}}\} \in E$ entails that the cover C must contain v_{i_k} or $v_{i_{k+1}}$, and it follows with $v_{i_k-1} \neq v_{i_k}$ *and* $v_{i_{k+1}-1} \neq v_{i_{k+1}}$ that the respective position in the run is (v_{i_k}, f) *or* $(v_{i_{k+1}}, f)$ (in other words: $x_{i_k} = f$ or $x_{i_{k+1}} = f$). Thus, the run contains infinitely many final states and is rejecting.

Thus, we have shown that $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ accepts the right set of trace-words. We now continue with the simpler proof that it accepts the right set of $\natural$-words.

First, words starting with $\natural v_0$ are accepted and in $\mathcal{L}'(\mathcal{G}_{v_0})$, while words starting with $\natural v$ and $v \neq v_0$ are rejected and not in $\mathcal{L}'(\mathcal{G}_{v_0})$.

A $\natural$-*word* that starts with $\alpha = v_1 v_2 v_3 \ldots v_n \natural w \in V^+ \natural V_\natural$ is in $\mathcal{L}'(\mathcal{G}_{v_0})$ if, and only if,

1. $v_{i-1} = v_i$ or $\{v_{i-1}, v_i\} \in E$ holds for all $i \leq n$, and
2. $v_n = w$.

If they both hold, the (accepting) run of $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ has the form $(v_0, n)(v_1, x_1)(v_2, x_2)(v_3, x_3)\ldots(v_n, x_n)(v_n, \natural)\top^\omega$.

If (1) holds but (2) does not, the (rejecting) run of $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ has the form $(v_0, n)(v_1, x_1)(v_2, x_2)(v_3, x_3)\ldots(v_n, x_n)(v_n, \natural)\bot^\omega$.

If (1) does not hold and $k \leq n$ is the smallest index with $v_{i-1} \neq v_i$ and $\{v_{i-1}, v_i\} \notin E$, the (rejecting) run of $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ has the form $(v_0, n)(v_1, x_1)(v_2, x_2)(v_3, x_3)\ldots(v_{k-1}, x_{k-1})\bot^\omega$.

As this covers all cases, we get $\mathcal{L}(\mathcal{B}_C^{\mathcal{G}_{v_0}}) = \mathcal{L}'(\mathcal{G}_{v_0})$. ◀

Corollary 12 and Lemma 13 immediately imply:

▶ **Corollary 14.** *Let C be a minimal vertex cover of a nice graph $\mathcal{G}_{v_0} = (V, E)$. Then $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ is a minimal deterministic Co-Büchi automaton that recognises the adjusted characteristic language of $\mathcal{G}_{v_0}$, and there is no good-for-games parity automaton with less states than $\mathcal{B}_C^{\mathcal{G}_{v_0}}$ that recognises the same language. Moreover, every minimal good-for-games automaton identifies a cover C' with $|C'| = |C|$.* ◀

The Corollaries 10 and 14 provide us with the hardness result.

▶ **Theorem 15.** *The following problems are NP hard.*

- *Given a good-for-games / deterministic Büchi automaton and a bound k, is there a language equivalent good-for-games Büchi automaton with at most k states / entries in its transition table (all 4 combinations)?*
- *Given a good-for-games / deterministic Co-Büchi automaton and a bound k, is there a language equivalent good-for-games Co-Büchi automaton with at most k states / entries in its transition table (all 4 combinations)?*
- *Given a good-for-games / deterministic parity / Büchi / Co-Büchi automaton and a bound k, is there a language equivalent good-for-games parity automaton with at most k states / entries in its transition table (all 12 combinations)?*

6 Discussion

We have established that determining if a good-for-games automaton with Büchi, Co-Büchi, or parity condition and state based acceptance is minimal, or that there is a GFG automaton with size up to k, is NP-complete. Moreover, this holds regardless of whether the starting automaton is given as a (Büchi, Co-Büchi, or parity) good-for-games automaton, or if it is presented as a (Büchi, Co-Büchi, or parity) deterministic automaton.

This drags three open questions into the limelight. The first is the complexity of testing whether or not a given nondeterministic automaton is good-for-games. Our results give no answer to this question: it simply accepts that a given automaton is good-for-games, and only guarantees a correct answer if the input is valid. GFG-ness is, however, known to be tractable for Büchi [1] and Co-Büchi [6] automata, and the extension to the more expressive class of parity good-for-games automata is active research.

It also raises the question if the difference is in good-for-games automata being inherently simpler to minimise, or if it is a property of choosing the less common transition based acceptance: the second open challenge is whether the tractability of minimising Co-Büchi good-for-games automata forebears the tractability of minimising the general class of parity good-for-games automata, while the third challenge is the question of whether NP hardness extends to transition based deterministic Büchi, Co-Büchi, and parity automata: the hard language used in this paper is not hard at all for transition based acceptance, as one can simply use final transitions between different v-states (and non-final self loops), cf. Figure 4. This could lend another argument for proliferating transition based acceptance.

In addition to the "transition vs. state based acceptance" question, another question is whether or not nondeterminism is the right starting point for GFG-ness, or if alternation is the better choice [2]. For such alternating automata, most of the succinctness and complexity questions for membership and minimisation are wide open.

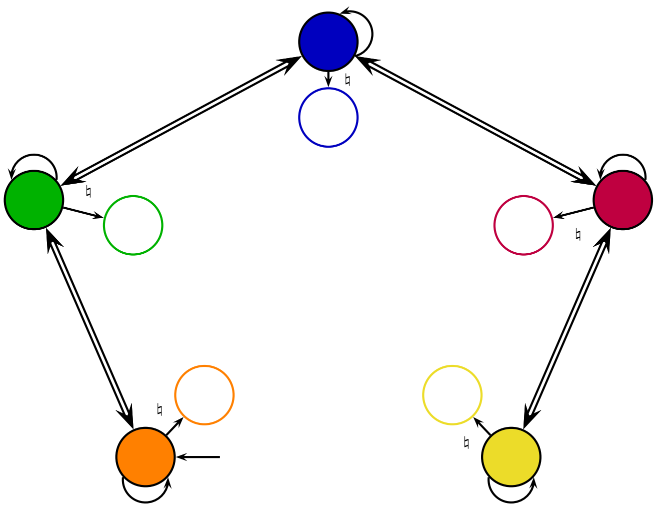

Figure 4 A minimal DBA with transition based acceptance for the running example.

References

1 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.16.

2 Udi Boker and Karoliina Lehtinen. Good for Games Automata: From Nondeterminism to Alternation. In Wan Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory (CONCUR 2019)*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CONCUR.2019.19.

3 E. Allen Emerson. Automata, tableaux and temporal logics. In *Proceedings of the International Conference on Logic of Programs (ICLP 1985), 17–19 June, Brooklyn, New York, USA*, volume 193 of *Lecture Notes in Computer Science*, pages 79–88. Springer-Verlag, 1985.

4 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006. doi:10.1007/11874683_26.

5 Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993. doi:10.1137/0222067.

6 Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.

7 Bader Abu Radi and Orna Kupferman. Minimizing GFG transition-based automata. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.100.

8 Sven Schewe. Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.400.

9 Sven Schewe. Minimising good-for-games automata is NP complete. *CoRR*, abs/2003.11979, 2020. arXiv:2003.11979.