


On the Minimisation of Deterministic and History-Deterministic Generalised (co)Büchi Automata

Antonio Casares   

LaBRI, Université de Bordeaux, France

University of Warsaw, Poland

Olivier Idir



IRIF, Université Paris-Cité, France

Denis Kuperberg   

LIP, CNRS, ENS Lyon, France

Corto Mascle  

LaBRI, Université de Bordeaux, France

Aditya Prakash  

University of Warwick, United Kingdom

Abstract

We present a polynomial-time algorithm minimising the number of states of history-deterministic generalised coBüchi automata, building on the work of Abu Radi and Kupferman on coBüchi automata. On the other hand, we establish that the minimisation problem for both deterministic and history-deterministic generalised Büchi automata is NP-complete, as well as the problem of minimising at the same time the number of states and colours of history-deterministic generalised coBüchi automata.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Automata minimisation, omega-regular languages, good-for-games automata

Digital Object Identifier [10.4230/LIPIcs.CVIT.2016.23](https://doi.org/10.4230/LIPIcs.CVIT.2016.23)

This document contains hyperlinks. On an electronic device, the reader can click on words or symbols (or just hover over them on some PDF readers) to see their definition.

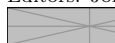
1 Introduction

First introduced by Büchi to obtain the decidability of monadic second order logic over $(\mathbb{N}, \text{succ})$ [13], automata over infinite words (also called ω -automata) have become a well-established area of study in Theoretical Computer Science. Part of its success is due to its applications to model checking (verify whether a system satisfies some given specifications) [5, 45, 25] and synthesis (given a set of specifications, automatically construct a system satisfying them) [12, 37]. In many of these applications, mainly in problems related to synthesis, non-deterministic models of automata are not well-suited, and costly determinisation procedures are usually needed [40].

In 2006, Henzinger and Piterman [27] proposed¹ a model of automata, called *history-deterministic* (HD)², presenting a restricted amount of non-determinism so that they exactly

¹ Similar ideas had been previously investigated by Kupferman, Safra and Vardi [32], and Colcombet studied history-determinism in the context of cost functions [21].

² These automata were first introduced under the name *good-for-games*. Currently, these two notions are no longer used interchangeably, although they coincide in the case of ω -automata. We refer to the



satisfy the properties that are needed for applications in synthesis. Namely, these automata do not need to guess the future: an automaton is history-deterministic if it admits a strategy resolving the non-determinism on the fly, in such a way that the run built by the strategy is accepting whenever the input word belongs to the language of the automaton. Since their introduction, several lines of research have focused on questions such as the succinctness of history-deterministic automata [31, 17], the problem of recognising them [4, 7], or extensions to other settings [33, 8, 9].

Minimisation of automata stands as one of the most fundamental problems in automata theory, for various reasons. Firstly, for its applications: when employing algorithms that rely on automata, having the smallest possible ones is crucial for efficiency. Secondly, beneath the problem of minimisation lies a profoundly fundamental question: What is the essential information needed to represent a formal language? A cornerstone result about automata over finite words is that each regular language admits a unique minimal deterministic automaton, in which states corresponds to the residuals of the language (the equivalence classes of the Myhill-Nerode congruence). Moreover, this minimal automaton can be obtained from an equivalent deterministic automaton with n states in time $\mathcal{O}(n \log n)$. [28].

However, the situation is quite different in the case of ω -automata. Contrary to the case of finite words, the residuals of a language are not sufficient to construct a correct deterministic automaton in general. In 2010, Schewe proved that the minimisation of deterministic Büchi automata is NP-complete [41]. That appeared to be a conclusion to the minimisation problem, but a crucial aspect of his proof was that the NP-completeness is established for automata with the acceptance condition *over states*, and this proof does not generalise to *transition-based* automata. A surprising positive result was obtained in 2019 by Abu Radi and Kupferman: we can minimise history-deterministic coBüchi automata using transition-based acceptance in polynomial time [1]. One year later, Schewe showed that the very same problem becomes NP-complete if state-based acceptance is used [42]. Multiple other results have backed the idea that transition-based acceptance is a better-suited model; we refer to [15, Chapter VI] for a detailed discussion. The work of Abu Radi and Kupferman raised the question of what is the complexity of the minimisation problem for other classes of transition-based automata such as (history-)deterministic Büchi automata. Since then, to the best of our knowledge, the only further result concerning minimisation of transition-based automata is Casares' NP-completeness proof for the problem of minimising deterministic Rabin automata [14].

In this paper, we focus our attention on generalised Büchi and generalised coBüchi automata, in which the acceptance condition is given, respectively, by conjunctions of clauses “see colour c infinitely often”, and by disjunctions of “eventually avoid colour d ”. Generalised (co)Büchi automata are as expressive as (co)Büchi automata, but they can be more succinct, due to their more complex acceptance condition. These automata appear naturally in the model-checking and synthesis of temporal properties [22, 26, 43]; for instance, SPOT's LTL-synthesis tool transforms a given LTL formula into a generalised Büchi automaton [23, 34]. Also, many efficient algorithms for their emptiness check have been developed [38, 39, 6].

Several works have approached the problem of reducing the state-space of generalised Büchi automata, which is usually done either by the use of simulations [43, 30] (which do not yield minimal automata), or by the application of SAT solvers [24, 3]. However, to the best of our knowledge, no theoretical result about the exact complexity of this minimisation problem appears in the literature.

survey [10] for further discussions.

Contributions

We provide a polynomial-time minimisation algorithm for history-deterministic generalised coBüchi automata (Theorem 11). Our algorithm uses Abu Radi-Kupferman’s minimal history-deterministic coBüchi automaton as a starting point, and reduces the state-space of this automaton in an optimal way to use a generalised coBüchi condition.

We prove that the minimisation problem is NP-complete for history-deterministic generalised Büchi automata (Theorem 28), as well as for deterministic generalised Büchi and generalised coBüchi automata (Theorem 29). We remark that both the NP-hardness and the NP-upper bound are challenging. Indeed, to obtain that the problem is in NP, we first need to prove that a minimal HD generalised Büchi automaton only uses a polynomial number of output colours. Additionally, we adapt a proof from [18] to show that minimising at the same time the number of states and colours is NP-complete for all the previous models, including history-deterministic generalised coBüchi automata (Theorem 30).

We summarise the results about the state-minimisation of transition-based automata in Table 1.

Condition Model	coBüchi	Büchi	generalised coBüchi	generalised Büchi
Deterministic	Unknown	Unknown	NP-complete (Theorem 29)	NP-complete (Theorem 29)
History-deterministic	PTIME [2]	Unknown	PTIME (Theorem 11)	NP-complete (Theorem 28)

■ **Table 1** Complexity of the minimisation problem for different types of transition-based automata.

We note that the PTIME complexity of recognising HD automata can be lifted from Büchi and coBüchi conditions to their generalised versions (Corollary 10). This result can be considered folklore, although we have not find it explicitly in the literature. In Appendix A, we also lift the characterisation based on the G_2 game from (co)Büchi automata to generalised ones (Theorem 37) (a similar remark was suggested in the conclusion of [7]).

State-minimality. In this paper, we primarily focus our attention on the minimisation of the number of *states* of the automata. In Theorem 30 we also consider the minimisation of both the number of states and colours of the *acceptance condition*. We highlight that the decision on how we measure the size of the automata is orthogonal to putting the acceptance condition over transitions.

The reader may wonder why we focus on these quantities and not, e.g., on the number of transitions. This choice, which is standard in the literature ([2, 41]), is justified by various reasons. First, the number of transitions of an automaton is polynomial in the number of states. Indeed, we can assume that there are not two transitions between two states over the same input letter (see [17, Prop.18]), therefore, $|\Delta| \leq |Q|^2|\Sigma|$. Maybe more importantly, in the case of automata over finite words, each state of the minimal automaton carry a precise information about the language it recognises: a residual of it. Ideally, a construction for a state-minimal automaton for an ω -regular language should lead to an understanding of the essential information necessary to represent it.

The interest of minimising both the number of states and the number of colours comes from the fact that the number of colours can be exponential on the number of states, but the size of the representation of the automaton is polynomial in the sum of these quantities.

2 Preliminaries

The disjoint union of two sets A, B is written $A \uplus B$. The *empty word* is denoted ε . For an infinite word $w \in \Sigma^\omega$, we denote $\text{Inf}(w)$ the set of letters occurring infinitely often in w .

2.1 Automata

We let Σ be a finite alphabet. An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_{\text{init}}, \Delta, \Gamma, \text{col}, W)$, where Q is its set of states, q_{init} its *initial state*, $\Delta \subseteq Q \times \Sigma \times Q$ its set of transitions, Γ its *output alphabet*, $\text{col}: \Delta \rightarrow \Gamma$ a *labelling with colours*, and $W \subseteq \Gamma^\omega$ its *acceptance condition*. A state q is called *reachable* if there exists a path from q_{init} to q . The *size* of an automaton is its number of states, written $|Q|$. We write $p \xrightarrow{a:c} q$ if $(p, a, q) \in \Delta$ and $\text{col}((p, a, q)) = c$.

A *run* ρ on an infinite word $w = a_1 a_2 \dots \in \Sigma^\omega$ is an infinite sequence of transitions $\rho = (q_0, a_1, q_1)(q_1, a_2, q_2)(q_2, a_3, q_3), \dots \in \Delta^\omega$ with $q_0 = q_{\text{init}}$. It is *accepting* if the infinite word $c_1 c_2 \dots \in \Gamma^\omega$ defined by $c_i = \text{col}(q_{i-1}, a_i, q_i)$, called the *output* of ρ , belongs to W .

The *language of an automaton* \mathcal{A} , denoted $\mathcal{L}(\mathcal{A})$, is the set of words that admit an accepting run. We say that two automata \mathcal{A} and \mathcal{B} over the same alphabet are *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

An automaton \mathcal{A} is *deterministic* (resp. *complete*) if, for all $(p, a) \in Q \times \Sigma$, there exists at most (resp. at least) one $q \in Q$ such that $(p, a, q) \in \Delta$. We note that if \mathcal{A} is deterministic, a word $w \in \Sigma^\omega$ admits at most one run in \mathcal{A} .

2.2 Acceptance conditions

In this paper we will focus on automata using generalised Büchi and generalised coBüchi acceptance conditions. A generalised Büchi condition can be seen as a conjunction of Büchi conditions, while a generalised coBüchi condition can be seen as a disjunction of coBüchi conditions.

A *generalised Büchi* condition with k colours is defined over the output alphabet $\Gamma = 2^C$, with C a set of k *output colours*, as

$$\text{genB}_C = \{w \in \Gamma^\omega \mid \bigcup \text{Inf}(w) = C\}.$$

It contains sequences of sets of colours such that every colour is seen infinitely often. Usually, we take $C = [k] = \{1, 2, \dots, k\}$.

The dual condition is the *generalised coBüchi* condition with k colours. That is, we define:

$$\text{genCoB}_C = \{w \in \Gamma^\omega \mid \bigcup \text{Inf}(w) \neq C\}.$$

It contains sequences of sets of colours such that at least one colour is seen finitely often.

The *size of the representation* of an automaton using a generalised (co)Büchi condition with k colours is $|Q| + |\Sigma| + k$; such an automaton can be described in polynomial space in this measure.

A *Büchi condition* (resp. *coBüchi condition*) can be defined as a generalised Büchi (resp. generalised coBüchi) condition in which $k = 1$. In this case, we call *Büchi transitions* (resp.

coBüchi transitions) the transitions $(p, a, q) \in \Delta$ such that $\text{col}((p, a, q)) = \{1\}$. An automaton using an acceptance condition of type X is called an *X-automaton*.

A language $L \subseteq \Sigma^\omega$ is *(co)Büchi recognisable* if there exists a deterministic (co)Büchi automaton \mathcal{A} such that \mathcal{A} recognises L . These coincide with languages recognised by generalised (co)Büchi automata (see Corollary 9). We note that non-deterministic (generalised) Büchi automata are strictly more expressive, while non-deterministic (generalised) coBüchi automata are as expressive as deterministic ones [35].

2.3 History-determinism

An automaton is called history-deterministic (or HD for short), if there exists a function, called a resolver, that resolves the non-determinism of \mathcal{A} depending only on the prefix of the input word read so far. Formally, a *resolver* for an automaton \mathcal{A} is a function $\sigma : \Sigma^+ \rightarrow \Delta$ such that for all words $w = a_0a_1 \dots \in \Sigma^\omega$, the sequence $\sigma^*(w) = \sigma(a_0)\sigma(a_0a_1)\sigma(a_0a_1a_2) \dots \in \Delta^\omega$ (called the *run induced by σ over w*) satisfies:

1. ρ is a run on w in \mathcal{A} ,
2. if $w \in \mathcal{L}(\mathcal{A})$, then ρ is an accepting run.

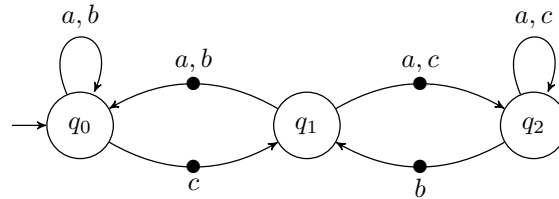
An automaton is *history-deterministic* if it admits a resolver. We say that a (deterministic/history-deterministic) automaton is *minimal* if it has a minimal number of states amongst equivalent (deterministic/history-deterministic) automata.

► **Remark 1.** Every deterministic automaton is HD. While the converse is false (see Example 2 below), we note that any language $L \subseteq \Sigma^\omega$ recognised by an HD Büchi automaton (resp. coBüchi automaton), can be recognised by a deterministic Büchi automaton (resp. deterministic coBüchi automaton) [32].

► **Example 2** (From [16, Ex. 2.3]). Let $\Sigma = \{a, b, c\}$ and $L = \{w \in \Sigma^\omega \mid \{b, c\} \not\subseteq \text{Inf}(w)\}$, that is, L is the set of words that contain either b or c only finitely often.

In Figure 1 we show an automaton recognising L that is not *determinisable by pruning*, that is, it cannot be made deterministic just by removing transitions.

We claim that this automaton is history-deterministic. First, we remark that the only non-deterministic choice appears when reading letter a from the state q_1 . A resolver can be defined as follows: whenever we have arrived to q_1 from q_0 (by reading letter c), if we are given letter a we go to state q_2 ; if we have arrived to q_1 from q_2 (by reading letter b), we will go to state q_0 . Therefore, if after some point letter b (resp. letter c) does not appear, we will stay forever in state q_2 (resp. state q_1) and accept.



■ **Figure 1** A history-deterministic coBüchi automaton recognising the language $L = \{w \in \Sigma^\omega \mid \{b, c\} \not\subseteq \text{Inf}(w)\}$. CoBüchi transitions are represented with a dot on them.

2.4 Residuals and prefix-independence

Let $L \subseteq \Sigma^\omega$ and $u \in \Sigma^*$. The *residual of L with respect to u* is the language

$$u^{-1}L = \{w \in \Sigma^\omega \mid uw \in L\}.$$

We write $[u]_L = \{v \in \Sigma^* \mid u^{-1}L = v^{-1}L\}$, and $\text{Res}(L)$ for the set of residuals of a language L .

Given an automaton \mathcal{A} and a state q , we denote \mathcal{A}^q the automaton obtained by setting q as initial state, and we refer to $\mathcal{L}(\mathcal{A}^q)$ as the *language recognised by q* . We say that two states q, p are *equivalent*, written $q \sim p$, if they recognise the same language. We note $[q]_{\mathcal{A}}$ the set of states equivalent to q (we simply write $[q]$ when \mathcal{A} is clear from the context).

We say that an automaton \mathcal{A} is *semantically deterministic* if non-deterministic choices lead to equivalent states, that is, if for every state q and pair of transitions $q \xrightarrow{a} p_1, q \xrightarrow{a} p_2$ we have $p_1 \sim p_2$.

If \mathcal{A} is semantically deterministic and $u \in \Sigma^*$ is a word labelling a path from the initial state to q , then $\mathcal{L}(\mathcal{A}^q) = u^{-1}L$. We say then that $u^{-1}L$ is the *residual associated to q* . For a residual $R \in \text{Res}(L)$ we denote Q^R the set of states of \mathcal{A} recognising R . We remark that $Q^R = [q]_{\mathcal{A}}$ for any state q recognising R .

We say that L is *prefix-independent* if for all $w \in \Sigma^\omega$ and $u \in \Sigma^*$, $w \in L \iff uw \in L$.

► **Remark 3.** A language L is prefix-independent if and only if it has a single residual.

2.5 Morphisms of automaton structures

An *automaton structure* over an alphabet Σ is a tuple $\mathcal{S} = (Q, \Delta)$, where $\Delta \subseteq Q \times \Sigma \times Q$. Let $\mathcal{S}_1 = (Q_1, \Delta_1)$ and $\mathcal{S}_2 = (Q_2, \Delta_2)$ be two automaton structures over the same alphabet. A *morphism of automaton structures* is a mappings $\phi: Q_1 \rightarrow Q_2$ such that for every $(q, a, q') \in \Delta_1$, $(\phi(q), a, \phi(q')) \in \Delta_2$. We note that such a morphism induces a function $\phi_\Delta: \Delta_1 \rightarrow \Delta_2$ sending (q, a, q') to $(\phi(q), a, \phi(q'))$. We also denote this function ϕ , whenever no confusion arises, and denote a morphism of automaton structures by $\phi: \mathcal{S}_1 \rightarrow \mathcal{S}_2$.

3 First properties and examples

We discuss a further example of a history-deterministic automaton and state some well-known facts about these automata that will be relevant for the rest of the paper.

3.1 A central example

The following automata will be used as a running example in Section 4.

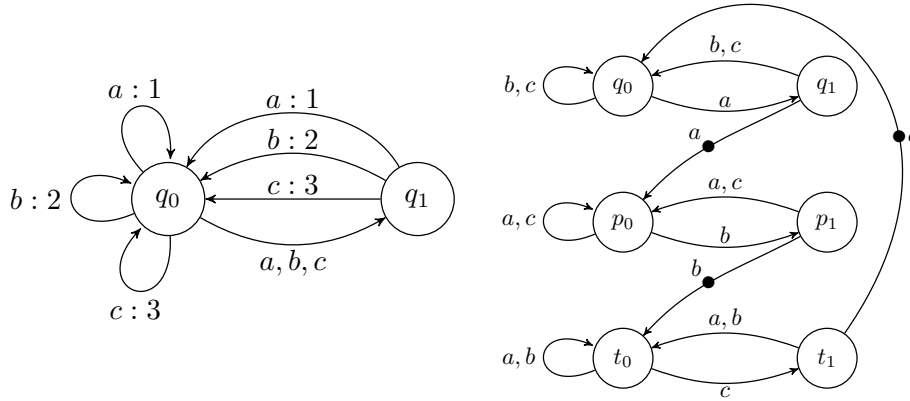
► **Example 4.** Let Σ_n be an alphabet of size n , and let

$$L_n = \{w \in \Sigma_n^\omega \mid \text{for some } x \in \Sigma_n \text{ the factor } xx \text{ appears only finitely often in } w\}.$$

On the left of Figure 2 we show a history-deterministic generalised coBüchi automaton recognising L_n with just 2 states (we show it for $\Sigma_3 = \{a, b, c\}$, but the construction clearly generalises to any n). The set of colours is $C = \{1, 2, 3\}$, and we accept if eventually some colour is not produced. A resolver can be defined as follows: in a round-robin fashion, we bet that the factor that does not appear is aa , then bb , then cc . While factor aa is not seen, we will take transition $q_0 \xrightarrow{a} q_1$ whenever letter a is read, to try to avoid colour 1. Whenever factor aa is read, we switch to the corresponding strategy with letter b , trying to avoid colour

2. If eventually factor xx is not produced, for $x \in \{a, b, c\}$, then some colour will forever be avoided.

We show a deterministic coBüchi automaton for L_3 on the right of Figure 2. Applying the characterisation of Abu Radi and Kupferman [2] (see Theorem 16), we can prove that this automaton is minimal amongst HD coBüchi automata. More generally, we can prove that a minimal HD coBüchi automaton for L_n has at least $2n$ states, and in fact, in this case, this optimal bound can be achieved with a deterministic automaton.



■ **Figure 2** On the left, a history-deterministic generalised coBüchi automaton recognising the language L_3 of words eventually avoiding factor xx for some letter x . On the right, a minimal deterministic coBüchi automaton for the same language (coBüchi transitions have a dot on them). In both cases, the initial state is irrelevant, as the language is prefix-independent.

3.2 Duality Büchi - coBüchi

► **Remark 5.** Let \mathcal{A} be a deterministic generalised Büchi automaton of size n and using k output colours. It suffices to replace the acceptance condition $\text{genB}_{[k]}$ with $\text{genCoB}_{[k]}$ to obtain a deterministic generalised coBüchi automaton of size n and using k output colours recognising the complement language $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$. Symmetrically, we can turn any deterministic generalised coBüchi automaton into a deterministic generalised Büchi automaton with the same number of states and colours recognising the complement language. As a consequence, the minimisations of deterministic generalised Büchi automaton and deterministic generalised coBüchi automaton are linear-time-equivalent problems.

We highlight that the hypothesis of determinism in the previous remark is crucial. This duality property no longer holds for non-deterministic (or history-deterministic) automata.

► **Lemma 6.** *There exists a history-deterministic generalised coBüchi automaton \mathcal{A} such that any history-deterministic generalised Büchi automaton recognising $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$ has strictly more states than \mathcal{A} .*

Such an example is provided by the language L_3 from Example 4 (in Lemma 45 we will prove that any non-deterministic generalised Büchi automaton recognising $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$ has at least 3 states). Relatedly, Kuperberg and Skrzypczak showed that the gap between an HD coBüchi and an HD Büchi automaton for the complement language can be exponential as well [31].

3.3 From generalised (co)Büchi to (co)Büchi

Deterministic coBüchi automaton for genCoB_C . Let $C = \{1, 2, \dots, k\}$ be a set of k colours; for convenience, we will use i to denote the colour $(i \bmod k)$, in particular, $k + 1 = 1$. We build a deterministic coBüchi automaton $\mathcal{D}_C^{\text{coB}}$ over the alphabet $\Gamma = 2^C$ recognising the language genCoB_C . It has as a state q_i for each colour $i \in C$ and contains the transitions $q_i \xrightarrow{X:\emptyset} q_i$, if $i \notin X$, and $q_i \xrightarrow{X:1} q_{i+1}$, if $i \in X$. The initial state is arbitrary.

We claim that the automaton $\mathcal{D}_C^{\text{coB}}$ recognises the language genCoB_C . First, we remark that the accepting runs of $\mathcal{D}_C^{\text{coB}}$ are exactly those that eventually remain forever in a state q_i . Let $w = w_1 w_2 \dots \in 2^C$. If w is accepted by $\mathcal{D}_C^{\text{coB}}$, then the run on w eventually stays in a q_i , so w eventually does not contain colour i , and $w \in \text{genCoB}_C$. Conversely, if w is rejected by $\mathcal{D}_C^{\text{coB}}$, it takes all transitions $q_i \xrightarrow{X:1} q_{i+1}$ infinitely often, so w must contain all colours in C infinitely often.

We define in a similar fashion a deterministic Büchi automaton \mathcal{D}_C^{B} recognising the language genB_C , simply by changing the acceptance condition of genCoB_C to genB_C .

► **Remark 7.** The automaton $\mathcal{D}_C^{\text{coB}}$ has k states, but exponentially many transitions.

Automata composition. Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma_{\mathcal{A}}, q_{\text{init}}^{\mathcal{A}}, \Delta_{\mathcal{A}}, \Gamma_{\mathcal{A}}, \text{col}_{\mathcal{A}}, W_{\mathcal{A}})$ and $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma_{\mathcal{B}}, q_{\text{init}}^{\mathcal{B}}, \Delta_{\mathcal{B}}, \Gamma_{\mathcal{B}}, \text{col}_{\mathcal{B}}, W_{\mathcal{B}})$ be two automata such that $\Sigma_{\mathcal{B}} = \Gamma_{\mathcal{A}}$ (i.e., \mathcal{B} is an automaton over the set of output colours of \mathcal{A}). The *composition* of \mathcal{A} and \mathcal{B} is the automaton over $\Sigma_{\mathcal{A}}$ defined as:

$$\mathcal{B} \circ \mathcal{A} = (Q_{\mathcal{A}} \times Q_{\mathcal{B}}, \Sigma_{\mathcal{A}}, (q_{\text{init}}^{\mathcal{A}}, q_{\text{init}}^{\mathcal{B}}), \Delta', \Gamma_{\mathcal{B}}, \text{col}', W_{\mathcal{B}}),$$

with transitions $(p_{\mathcal{A}}, p_{\mathcal{B}}) \xrightarrow{a:c} (q_{\mathcal{A}}, q_{\mathcal{B}})$ if $p_{\mathcal{A}} \xrightarrow{a:b} q_{\mathcal{A}} \in \Delta_{\mathcal{A}}$ and $p_{\mathcal{B}} \xrightarrow{b:c} q_{\mathcal{B}} \in \Delta_{\mathcal{B}}$.

Intuitively, given a word in $\Sigma_{\mathcal{A}}^{\omega}$, we feed the output of $\text{col}_{\mathcal{A}}$ directly as input to \mathcal{B} , while keeping track of the progression in both automata. We accept according to the acceptance condition of \mathcal{B} .

► **Lemma 8 (Folklore).** *Let \mathcal{A} be an automaton with acceptance condition $W \subseteq \Gamma^{\omega}$, and let \mathcal{B} be a deterministic automaton over Γ recognising W . Then $\mathcal{B} \circ \mathcal{A}$ recognises $\mathcal{L}(\mathcal{A})$, and the automaton $\mathcal{B} \circ \mathcal{A}$ is history-deterministic (resp. deterministic) if and only if \mathcal{A} is.*

Thus, to convert a generalised coBüchi automaton \mathcal{A} to an equivalent coBüchi one, we can just compose it with $\mathcal{D}_C^{\text{coB}}$. The symmetric result holds for generalised Büchi automata.

► **Corollary 9 (Folklore).** *Let \mathcal{A} be a generalised coBüchi automaton using C as output colours. The automaton $\mathcal{D}_C^{\text{coB}} \circ \mathcal{A}$ is a coBüchi automaton equivalent to \mathcal{A} which is (history-)deterministic if and only if \mathcal{A} is. Moreover, $\mathcal{D}_C^{\text{coB}} \circ \mathcal{A}$ can be computed in polynomial time in the size of the representation of \mathcal{A} . The same is true for generalised Büchi automata.*

We note that $\mathcal{D}_C^{\text{coB}} \circ \mathcal{A}$ has $k \cdot |\mathcal{A}|$ states, where $k = |C|$, but it might have exponentially many transitions in k . However, we underline that we can compute $\mathcal{D}_C^{\text{coB}} \circ \mathcal{A}$ from \mathcal{A} in polynomial time in the size of its representation. Indeed, we just need to compose \mathcal{A} with the restriction of $\mathcal{D}_C^{\text{coB}}$ to transitions whose letters are subsets of colours that appear in \mathcal{A} .

Deciding history-determinism. The problem of deciding whether an automaton is HD is known to be in PTIME for Büchi and coBüchi automata [31, 4, 7]. Combining this fact with Corollary 9, we directly obtain:

► **Corollary 10.** *Given a generalised Büchi (resp. generalised coBüchi) automaton, it is in PTIME to decide whether it is history-deterministic.*

A different proof of Corollary 10, which goes through the G_2 game [4], is presented in Appendix A.

4 Polynomial-time minimisation of HD generalised coBüchi automata

In this section we present one of the main contributions of the paper (Theorem 11): history-deterministic generalised coBüchi automata can be minimised in polynomial time.

► **Theorem 11.** *Given a history-deterministic generalised coBüchi automaton, we can build in polynomial time in its representation an equivalent history-deterministic generalised coBüchi automaton with a minimal number of states.*

The proof of this result strongly relies on the construction of minimal coBüchi automata by Abu Radi and Kupferman [2]. We will show that, for a coBüchi recognisable language L , we can extract a minimal HD generalised coBüchi automaton for L from its minimal HD coBüchi automaton.

In Section 4.1 we introduce some terminology and state the main property satisfied by the minimal HD coBüchi automaton of Abu Radi and Kupferman. We then present our construction, decomposing it in two steps for simplicity: first we construct a minimal HD generalised coBüchi automaton in the case of prefix-independent languages in Section 4.2, and in Section 4.3, we show how to get rid of the prefix-independence assumption.

4.1 Minimisation of HD coBüchi automata

Safe components and safe languages. A path $q \rightsquigarrow q'$ in a coBüchi automaton is *safe* if no coBüchi transition appears on it. Let $\mathcal{A}_{\text{safe}}$ be the automaton obtained by removing from \mathcal{A} all coBüchi transitions. A *safe component* of \mathcal{A} is a strongly connected component (i.e., a maximal set of states which are all reachable from each other) of $\mathcal{A}_{\text{safe}}$; formally, this is an automaton structure $\mathcal{S} = (S, \Delta_{\mathcal{S}})$ with $S \subseteq Q_{\mathcal{A}}$ and $\Delta_{\mathcal{S}} \subseteq \Delta_{\mathcal{A}}$. We let $\text{Safe}(\mathcal{A})$ be the set of safe components of \mathcal{A} .

We define the *safe language of a state q* as:

$$\mathcal{L}_{\text{Safe}}(\mathcal{A}^q) = \{w \in \Sigma^\omega \mid \text{there is a run } q \rightsquigarrow^w \text{ in } \mathcal{A}_{\text{safe}}\}.$$

► **Example 12.** The safe components of the automaton on the right of Figure 2 (page 7) have as set of states: $S_1 = \{q_0, q_1\}$, $S_2 = \{p_0, p_1\}$ and $S_3 = \{t_0, t_1\}$. The safe language of q_0 is $\mathcal{L}_{\text{Safe}}(\mathcal{A}^{q_0}) = \{w \in \Sigma^\omega \mid w \text{ does not contain the factor } aa\}$.

The following statement simply follows from the fact that the accepting runs of a coBüchi automaton are exactly those that eventually stay in a safe component.

► **Lemma 13.** *Let \mathcal{A} be a semantically deterministic coBüchi automaton. Let $w \in \Sigma^\omega$ be a word labelling a path in a safe component of \mathcal{A} starting from a state q . Then, $uw \in L$ for all $u \in \Sigma^*$ such that $q_{\text{init}} \rightsquigarrow^u q$.*

Nice coBüchi automata. We say that a coBüchi automaton \mathcal{A} is in *normal form* if all transitions between two different safe components are coBüchi transitions. We note that any coBüchi automaton can be put in normal form without modifying its language by setting all transitions between two different safe components to be coBüchi. We say that \mathcal{A} is *safe deterministic* if $\mathcal{A}_{\text{safe}}$ is a deterministic automaton. That is, if the non-determinism of \mathcal{A} appears exclusively in coBüchi transitions. We say that \mathcal{A} is *nice* if all its states are reachable, it is semantically deterministic, in normal form, and safe deterministic.

It is not difficult to see that any history-deterministic automaton can be assumed to be semantically deterministic (different choices made by a resolver from the same state must be

consistent with the residual). Kuperberg and Skrzypczak showed the more involved result that we can moreover assume safe determinism [31]. All in all, we have:

► **Lemma 14** ([31]). *Every history-deterministic coBüchi automaton \mathcal{A} can be turned in polynomial time into an equivalent nice HD coBüchi automaton $\mathcal{A}_{\text{nice}}$ such that:*

- $|\mathcal{A}_{\text{nice}}| \leq |\mathcal{A}|$,
- For every safe component \mathcal{S} of $\mathcal{A}_{\text{nice}}$, there is some safe component \mathcal{S}' of \mathcal{A} with $|\mathcal{S}| \leq |\mathcal{S}'|$.

Although the second item of the previous proposition is not explicitly stated in [31], it simply follows from the fact that all the transformations used to turn \mathcal{A} into a nice automaton either add coBüchi transitions to \mathcal{A} or remove transitions from it. These operations can only subdivide safe components.

Minimal HD coBüchi automata. We present the necessary conditions for the minimality of history-deterministic coBüchi automata identified by Abu Radi and Kupferman [2].

We say that a coBüchi automaton \mathcal{A} is *safe centralised* if for all equivalent states $q \sim p$, if the safe languages of q and p are comparable for the inclusion relation ($\mathcal{L}_{\text{Safe}}(\mathcal{A}^q) \subseteq \mathcal{L}_{\text{Safe}}(\mathcal{A}^p)$ or vice versa), then they are in the same safe component of \mathcal{A} . It is *safe minimal* if for all states $q \sim p$, the equality $\mathcal{L}_{\text{Safe}}(\mathcal{A}^q) = \mathcal{L}_{\text{Safe}}(\mathcal{A}^p)$ implies $q = p$.

► **Example 15.** The automaton on the right of Figure 2 is safe minimal and safe centralised. However, the automaton from Figure 1 is not safe centralised, as $\mathcal{L}_{\text{Safe}}(\mathcal{A}^{q_1}) = \emptyset \subseteq \mathcal{L}_{\text{Safe}}(\mathcal{A}^{q_2})$, but q_1 and q_2 appear in different safe components.

► **Theorem 16** ([2, Lemma 3.5]). *Let \mathcal{A}_{min} be a nice, safe minimal and safe centralised HD coBüchi automaton. Then, for any equivalent nice HD coBüchi automaton \mathcal{A} there is an injection $\eta: \text{Safe}(\mathcal{A}_{\text{min}}) \rightarrow \text{Safe}(\mathcal{A})$ such that for every safe component $\mathcal{S} \in \text{Safe}(\mathcal{A}_{\text{min}})$, it holds that $|\mathcal{S}| \leq |\eta(\mathcal{S})|$.*

It follows that nice safe minimal and safe centralised HD coBüchi automata have a minimal number of states.

► **Theorem 17** ([2, Theorem 3.15]). *Any coBüchi recognisable language L can be recognised by a nice, safe minimal and safe centralised HD coBüchi automaton $\mathcal{A}_L^{\text{coB}}$. Moreover, such an automaton $\mathcal{A}_L^{\text{coB}}$ can be computed in polynomial time from any HD coBüchi automaton recognising L .*

4.2 Minimal HD generalised coBüchi automata: prefix-independent case

In this subsection, we show how to minimise history-deterministic generalised coBüchi automata recognising prefix-independent languages. The prefix-independence hypothesis will be removed in the next subsection.

Let $L \subseteq \Sigma^\omega$ be a prefix-independent coBüchi recognisable language, and let \mathcal{A} be a history-deterministic generalised coBüchi automaton recognising it.

Combining Corollary 9 and Theorem 16, we obtain that we can build in polynomial time the minimal HD coBüchi automaton $\mathcal{A}_L^{\text{coB}}$ for L . Let $\text{Safe}(\mathcal{A}_L^{\text{coB}}) = \{\mathcal{S}_1, \dots, \mathcal{S}_k\}$ be an enumeration of the safe components of $\mathcal{A}_L^{\text{coB}}$, with S_i and Δ_i the sets of states and transitions of each safe component, respectively. We show how to build an HD generalised coBüchi automaton $\mathcal{A}_L^{\text{genCoB}}$ of size $n_{\text{max}} = \max_{1 \leq i \leq k} |S_i|$ and using k output colours.

Intuitively, $\mathcal{A}_L^{\text{genCoB}}$ will be the full automaton (it contains transitions between all pairs of states, for all input letters). Since $|S_i| \leq n_{\text{max}}$, we can map each safe component \mathcal{S}_i to this

full automaton via a morphism ϕ_i , and use (the non-appearance of) colour i to accept runs that eventually would have stayed in the safe component \mathcal{S}_i in $\mathcal{A}_L^{\text{coB}}$. That is, the transitions of $\mathcal{A}_L^{\text{genCoB}}$ that are “safe-for-colour i ” will be exactly those in $\phi_i(\mathcal{S}_i)$.

Formally, let $\mathcal{A}_L^{\text{genCoB}} = (Q, \Sigma, q_{\text{init}}, \Delta, \Gamma, \text{col}, \text{genCoB})$ with:

- $Q = \{p_1, p_2, \dots, p_{n_{\text{max}}}\}$,
- $q_{\text{init}} = p_1$ (any state can be chosen as initial),
- $\Delta = Q \times \Sigma \times Q$,
- $\Gamma = 2^{\{1, \dots, k\}}$.

Finally, we define the colour labelling $\text{col}: \Delta \rightarrow \Gamma$. For each $1 \leq i \leq k$, let $\phi_i: \mathcal{S}_i \rightarrow (Q, \Delta)$ be any injective morphism of automaton structures (such a morphism exists since $|\mathcal{S}_i| \leq n_{\text{max}}$ and (Q, Δ) is the full automaton structure). We put colour i in a transition $e \in \Delta$ if and only if there is no transition $e' \in \Delta_i$ such that $\phi_i(e') = e$. That is, $\text{col}(e) = \{i \mid \phi_i^{-1}(e) = \emptyset\}$.

► **Remark 18.** We remark that this colour labelling uses some arbitrary choices, namely, the way we map the different safe components of $\mathcal{A}_L^{\text{coB}}$ to the full automaton of size n_{max} . In particular, there is no unique minimal HD generalised coBüchi automaton recognising L . By a slight abuse of notation, we denote $\mathcal{A}_L^{\text{genCoB}}$ one automaton originated by this procedure.

► **Example 19.** The automaton on the left of Figure 2 (page 7) (almost) corresponds to this construction. Indeed, it has been obtained by assigning a colour to each safe component of $\mathcal{A}_L^{\text{coB}}$ (on the right) and superposing them in a 2-state automaton. To simplify its presentation, we have removed some unnecessary transitions of $\mathcal{A}_L^{\text{genCoB}}$, that is why the automaton displayed is not the full-automaton.

► **Proposition 20 (Correctness).** *Let L be a prefix-independent language that is coBüchi recognisable. The automaton $\mathcal{A}_L^{\text{genCoB}}$ is history-deterministic and recognises L .*

Proof sketch. If w admits an accepting run ρ in $\mathcal{A}_L^{\text{genCoB}}$, then its run eventually does not produce some colour i in its output. This means that, eventually, such a run is the ϕ_i -projection of a run in a safe component of $\mathcal{A}_L^{\text{coB}}$, so $w \in L$.

A resolver for $\mathcal{A}_L^{\text{genCoB}}$ can be defined as follows: in a round-robin fashion we follow the different safe components of $\mathcal{A}_L^{\text{coB}}$. If a colour i is produced while we are trying to avoid it, we go back to p_1 and try to avoid colour $i' = (i + 1) \bmod k$ by following the safe component $\mathcal{S}_{i'}$. If a word w belongs to L , it eventually admits a safe path in $\mathcal{A}_L^{\text{coB}}$, so it will be accepted by this resolver. ◀

► **Proposition 21 (Minimality).** *Let \mathcal{B} be a history-deterministic generalised coBüchi automaton recognising a prefix-independent language L . Then, $|\mathcal{A}_L^{\text{genCoB}}| \leq |\mathcal{B}|$.*

Proof. Let $C = \{1, \dots, k\}$ be the set of output colours used by the acceptance condition of \mathcal{B} and let $\mathcal{D}_C^{\text{coB}}$ be the coBüchi automaton recognising genB_C presented in Section 3.3. By Corollary 9, $\mathcal{D}_C^{\text{coB}} \circ \mathcal{B}$ is a history-deterministic automaton recognising L . Moreover, the states of $\mathcal{D}_C^{\text{coB}} \circ \mathcal{B}$ are a disjoint union $Q_1 \uplus Q_2 \uplus \dots \uplus Q_k$ such that:

- $|Q_i| = |\mathcal{B}|$,
- all transitions leaving Q_i are coBüchi transitions going to Q_{i+1} , where $Q_{k+1} = Q_1$.

Therefore, each safe component \mathcal{S} of $\mathcal{D}_C^{\text{coB}} \circ \mathcal{B}$ is included in some Q_i , so $|\mathcal{S}| \leq |\mathcal{B}|$. By Lemma 14, we can turn $\mathcal{D}_C^{\text{coB}} \circ \mathcal{B}$ into a nice HD coBüchi automaton \mathcal{B}' satisfying that none of its safe components is larger than $|\mathcal{B}|$.

By Theorem 16 there is an injection $\eta: \text{Safe}(\mathcal{A}_L^{\text{coB}}) \rightarrow \text{Safe}(\mathcal{B}')$ such that $|\mathcal{S}| \leq |\eta(\mathcal{S})|$ for all safe component \mathcal{S} of $\mathcal{A}_L^{\text{coB}}$. In particular, if \mathcal{S}_{max} is a safe component of maximal size in $\mathcal{A}_L^{\text{coB}}$, we obtain: $|\mathcal{A}_L^{\text{genCoB}}| = |\mathcal{S}_{\text{max}}| \leq |\eta(\mathcal{S}_{\text{max}})| \leq |\mathcal{B}|$. ◀

4.3 Minimal HD generalised coBüchi automata: general case

We now describe the polynomial-time construction for minimising a given HD generalised coBüchi automaton (without any prefix-independence assumptions). For the optimality proof, we can reduce to the simplest prefix-independent case.

We fix an HD generalised coBüchi automaton \mathcal{A} recognising a language L . As before, using Corollary 9 and the minimisation procedure of Abu Radi and Kupferman, we can obtain the minimal HD coBüchi automaton $\mathcal{A}_L^{\text{coB}}$ in polynomial time. We show how to convert it to an equivalent HD generalised coBüchi automaton $\mathcal{A}_L^{\text{genCoB}}$ of minimal size.

Let R_1, R_2, \dots, R_m be all the distinct residual languages of L , i.e., languages of the form $u^{-1}L$ for some finite word $u \in \Sigma^*$. We note that these residuals induce a partition of the states of $\mathcal{A}_L^{\text{coB}}$ into Q^{R_1}, \dots, Q^{R_m} , where the states in Q^{R_j} recognise R_j . We assume that $R_1 = L$ is the residual corresponding to the initial state of $\mathcal{A}_L^{\text{coB}}$. Let $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ be the safe components of $\mathcal{A}_L^{\text{coB}}$, with S_i and Δ_i as sets of states and transitions, respectively. For each residual language R_j , define n_j as the largest number of states recognising R_j appearing in a safe component of $\mathcal{A}_L^{\text{coB}}$. That is,

$$n_j = \max_{1 \leq i \leq k} |S_i \cap Q^{R_j}|.$$

We shall construct a language-equivalent HD generalised coBüchi automaton $\mathcal{A}_L^{\text{genCoB}}$ with $n_1 + n_2 + \dots + n_m$ states. Towards this, for each residual language R_j , let $P_j = \{p_j^1, p_j^2, \dots, p_j^{n_j}\}$ be a set of n_j elements. The automaton $\mathcal{A}_L^{\text{genCoB}} = (Q, \Sigma, q_{\text{init}}, \Delta, \Gamma, \text{col}, \text{genB})$ is given by:

- $Q = P_1 \uplus P_2 \uplus \dots \uplus P_m$.
- $q_{\text{init}} = p_1^1$ (any state corresponding to the residual of the initial state of $\mathcal{A}_L^{\text{coB}}$ would work).
- Let (q, a, q') be a transition in $\mathcal{A}_L^{\text{coB}}$, with $q \in Q^{R_j}$ and $q' \in Q^{R_{j'}}$. Then, $(p, a, p') \in \Delta$ for all $p \in P_j$ and $p' \in P_{j'}$.
- $\Gamma = 2^{\{1, \dots, k\}}$.

One way of picturing $\mathcal{A}_L^{\text{genCoB}}$ is by taking the automaton of residuals of L and making n_j copies of the state corresponding to each residual R_j (while keeping all transitions).

We now describe the colour labelling $\text{col}: \Delta \rightarrow \Gamma$. Informally, each safe component \mathcal{S}_i is mapped into $\mathcal{A}_L^{\text{genCoB}}$ so that the states of $S_i \cap R_j$ are mapped into P_j . These safe components are then “superimposed” upon each other and coloured appropriately, so that a run eventually staying in \mathcal{S}_i in $\mathcal{A}_L^{\text{coB}}$ corresponds to a run in $\mathcal{A}_L^{\text{genCoB}}$ that eventually avoids colour i .

More formally, for $i \in [1, k]$, let $\phi_i: \mathcal{S}_i \rightarrow \mathcal{A}_L^{\text{genCoB}}$ be an injective morphism such that $\phi_i(q) \in P_j$ if $q \in Q^{R_j}$. Such injective morphism does indeed exist, by the choice of n_j and the fact that $\mathcal{A}_L^{\text{genCoB}}$ contains all transitions consistent with the residuals. The transitions of Δ that are i -safe are defined to be exactly those that are the image by ϕ_i of some transition in \mathcal{S}_i . That is, for $e \in \Delta$, the labelling $\text{col}(e)$ contains exactly the colours in $\{i \mid \phi_i^{-1}(e) = \emptyset\}$.

► **Remark 22.** We remark that the automaton $\mathcal{A}_L^{\text{genCoB}}$ obtained in this way uses a polynomial number of output colours in the size of the minimal HD coBüchi automaton $\mathcal{A}_L^{\text{coB}}$ (see also Lemma 42). However, the number of colours used is not necessarily optimal (see Theorem 30).

The correctness of our construction, stated below, is proven similarly to Proposition 20.

► **Proposition 23 (Correctness).** *Let L be a coBüchi recognisable language. The automaton $\mathcal{A}_L^{\text{genCoB}}$ is history-deterministic and recognises L .*

We explain how to obtain the minimality of $\mathcal{A}_L^{\text{genCoB}}$, stated below. We reduce to the prefix-independent case, using a technique from [11].³ Full proofs can be found in Appendix B.

► **Proposition 24** (Minimality). *Let \mathcal{B} be a history-deterministic generalised coBüchi automaton recognising a language L . Then, $|\mathcal{A}_L^{\text{genCoB}}| \leq |\mathcal{B}|$.*

For each residual $R = u^{-1}L \in \text{Res}(L)$, we define the *local alphabet at R* , as:

$$\Sigma|_R = \{v \in \Sigma^+ \mid [uv] = [u] \text{ and for any proper prefix } v' \text{ of } v, [uv'] \neq [v]\}.$$

That is, if \mathcal{A} is semantically deterministic, then $\Sigma|_R$ is the set of words that connect states in Q^R . Note that in general $\Sigma|_R$ may be infinite, however this is harmless in this context, and we will freely allow ourselves to talk about automata over infinite alphabets. Also, $\Sigma|_R$ is empty if and only if all the states of Q^R are *transient*, that is, they do not occur in any cycle of the automaton. For simplicity, in the following we assume that no state of \mathcal{A} is transient; the general case is treated in detail in Appendix B.

We define the *localisation of L to a residual $R \in \text{Res}(L)$* as the language over the alphabet $\Sigma|_R$ given by: $L|_R = \{w \in \Sigma|_R^\omega \mid w \in R\}$.

► **Remark 25.** For every residual R , $L|_R$ is a prefix-independent language.

Let \mathcal{A} be a semantically deterministic generalised coBüchi automaton with k colours recognising $L \subseteq \Sigma^\omega$. For each recurrent residual R of L we define $\mathcal{A}|_R$ to be the generalised coBüchi automaton over $\Sigma|_R$ given by:

- the set of states is Q^R , that is, the set of states of \mathcal{A} recognising R .
- the initial state is arbitrary,
- the acceptance condition is genCoB_C (for C the output colours of \mathcal{A}),
- there is a transition $q \xrightarrow{w:X} p$, with $w \in \Sigma|_R$, $X \in 2^{\{1,\dots,k\}}$, if there is a path from q to p labelled w and producing the set of colours X in \mathcal{A} .

► **Lemma 26.** *The automaton $(\mathcal{A}|_R)^q$ recognises $L|_R$ for each $q \in Q^R$. Moreover, if \mathcal{A}^q is history-deterministic, so is $(\mathcal{A}|_R)^q$.*

Using the fact that (safe) paths between states in Q^R are the same in \mathcal{A} or in $\mathcal{A}|_R$, combined with Theorem 16, we can prove:

► **Lemma 27.** *$\mathcal{A}_L^{\text{coB}}|_R$ is a minimal HD coBüchi automaton recognising $L|_R$. Moreover, a maximal safe component of this automaton has size n_j .*

To conclude the proof of Proposition 24, we combine Lemma 27 with Proposition 21 to show that $|Q_{\mathcal{B}}^{R_j}| \geq n_j$. This implies: $|\mathcal{B}| \geq n_1 + \dots + n_m = |\mathcal{A}_L^{\text{genCoB}}|$.

5 NP-completeness of minimisation of deterministic and HD generalised Büchi automata

In this section we contrast the polynomial-time complexity previously obtained for minimising history-deterministic generalised coBüchi automata with the NP-hardness of the minimisation of deterministic generalised (co)Büchi and history-deterministic generalised Büchi automata.

³ An alternative proof scheme is to extend the proof of Proposition 21 to the general case, by taking into account the residuals of the language. For this, we need a refinement of Theorem 16, stating that the injection η satisfies that, for every residual R and safe component \mathcal{S} of $\mathcal{A}_L^{\text{coB}}$, $|\mathcal{S} \cap R| \leq |\eta(\mathcal{S}) \cap R|$. The proof of Abu Radi and Kupferman [2] does indeed lead to this result, but it is not explicitly stated in this form.

► **Theorem 28.** *The following problem is NP-complete: Given a history-deterministic generalised Büchi automaton \mathcal{A} and a number n , decide whether there is an equivalent history-deterministic generalised Büchi automaton with at most n states.*

► **Theorem 29.** *The minimisation of the number of states of deterministic generalised Büchi and generalised coBüchi automata is NP-complete.*

We show NP-hardness of the minimisation problems in the deterministic and history-deterministic Büchi cases simultaneously. The NP-hardness for the deterministic coBüchi case follows directly. Our reduction is from a suitable version of the 3-colouring problem.

We further consider the problem of minimising both colours and states simultaneously for generalised (co)Büchi automata. For the automata classes appearing in the previous theorems (deterministic and history-deterministic generalised Büchi automata), it easily follows that this problem is NP-complete. We focus therefore in the case of history-deterministic generalised coBüchi, for which the minimisation of states has proven to be polynomial (Theorem 11). We show that, even in this case, minimising both states and colours is NP-complete.

► **Theorem 30.** *The following problem is NP-complete: Given a history-deterministic generalised coBüchi automaton \mathcal{A} , and numbers n and k , decide whether there is an equivalent history-deterministic generalised coBüchi automaton with at most n states and k colours.*

We obtain the lower bound by adapting the proof of NP-hardness of the minimising of Rabin pairs, given in [18], itself inspired from [29]. Details can be found in Appendix C.3.

5.1 Containment in NP and bounds on the necessary number of colours

In this section we address an important subtlety of our minimisation problems: We minimise the number of states, but the number of colours used by a generalised Büchi or generalised coBüchi automaton may be exponential in its number of states.

In order to show that the problems at hand are in NP, we need to show that the minimal deterministic and history-deterministic automata require a number of colours that is polynomial in the size of the input (that is, the number of states and colours of the input automaton). This turns out to be true, although not trivial. Full proofs are in Appendix C.1.

► **Lemma 31.** *Let \mathcal{A} be a deterministic generalised Büchi automaton with n states and k colours. Then there is an equivalent deterministic generalised Büchi automaton with a minimal number of states and using $O(n^2k)$ colours.*

► **Lemma 32.** *Let \mathcal{A} be a history-deterministic generalised Büchi automaton with n states and k colours. Then there is an equivalent history-deterministic generalised Büchi automaton with a minimal number of states and using $O(n^3k^2)$ colours.*

This allows us to obtain an NP algorithm as we only need to guess an automaton with polynomially many states and colours, and check equivalence with the input automaton. The latter test can be done in polynomial time (see for instance [42, Thm. 4]).

As an additional result, we show that the previous lemmas do not hold for general non-deterministic automata: minimising an automaton may blow up its number of colours.

► **Proposition 33.** *There exists a family of non-deterministic generalised Büchi automata $(\mathcal{A}_n)_{n \in \mathbb{N}}$ such that for all n , \mathcal{A}_n uses $n + 1$ states and 2 colours and a minimal automaton equivalent to \mathcal{A}_n requires 2^n colours.*

5.2 Hardness of state minimisation

We provide a reduction from the 3-colouring problem. We construct from a given graph G a deterministic automaton \mathcal{A}_G such that:

- If G is 3-colourable then there is a 3-state deterministic automaton equivalent to \mathcal{A} , and
- if G is not 3-colourable then there is no automaton \mathcal{B} (deterministic or not) with 3 states equivalent to \mathcal{A} .

This establishes the hardness of state-minimisation for deterministic and history-deterministic automata simultaneously. The full proof is in Appendix C.2. We only present here the languages we use and a sketch of the first item. The second item is obtained by a refined case analysis over the cycles of generalised Büchi automata with three states.

Given an undirected graph $G = (V, E)$, we define the *neighbourhood* of a vertex v as the set $N[v] = \{v' \in V \mid \{v, v'\} \in E\}$, and its *strict neighbourhood* as $n(v) = N[v] \setminus \{v\}$.

We consider the alphabet $\Sigma = V$. For each $v \in V$, we define the language:

$$L_v = (V^*vv)^\omega \cup (V^*(V \setminus N[v]))^\omega \quad \text{and we let} \quad L_G = \bigcap_{v \in V} L_v.$$

In words, a sequence of nodes is in L_G if for all $v \in V$ it either has infinitely many factors vv or sees a vertex that is not a neighbour of v infinitely many times.

The first item is proven by the following more general lemma. We actually show that from a k -colouring of G we can build a deterministic automaton with k states for L_G . This also allows us to construct the automaton \mathcal{A}_G by applying this lemma on a trivial $|V|$ -colouring.

► **Lemma 34.** *For all graph $G = (V, E)$ and $k \in \mathbb{N}$, if G is k -colourable then there exists a complete deterministic generalised Büchi automaton \mathcal{B} with k states which recognises L_G .*

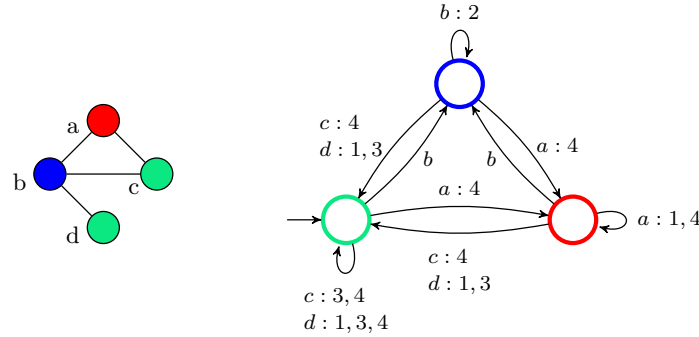
Proof sketch. Suppose G is k -colourable, let $c : V \rightarrow \{1, \dots, k\}$ be a k -colouring of G . We define the deterministic generalised Büchi automaton \mathcal{B} as follows:

- The set of states is $Q = \{1, \dots, k\}$, we pick any state as the initial one.
- For $q \in Q$ and $v \in \Sigma = V$, the v -transition from q is $q \xrightarrow{v} c(v)$.
- The set of output colours is V , hence the output alphabet is $\Gamma = 2^V$.
- If $q \neq c(v)$, the transition $q \xrightarrow{v} c(v)$ is coloured with $V \setminus N[v]$. Transitions of the form $c(v) \xrightarrow{v} c(v)$ are coloured with $V \setminus n(v)$.

It is then quite straightforward to show that a word is accepted by \mathcal{B} if and only if for each v it goes infinitely many times through the v -loop on $c(v)$ or sees infinitely many times vertices outside of $N[v]$. The structure of the automaton ensures that those words are exactly the ones in L_v . In particular, the fact that $c(u) \neq c(v)$ for all neighbours u and v implies that we cannot go through the v loop on $c(v)$ without reading a v or a non-neighbour of v just before. Figure 3 shows an example of this construction. ◀

6 Conclusion

We believe that one of the key novel insights of this work is to compare the complexity of the minimisation of HD generalised coBüchi automata (polynomial) with both HD generalised Büchi automata and deterministic models (NP-complete). For history-deterministic and deterministic Büchi automata the minimisation problem is still open; our results are an important step in this direction, and seem to indicate that the polynomial-time minimisation algorithm for the HD coBüchi case will not extend to the Büchi or the deterministic case.



■ **Figure 3** A graph with a 3-colouring, and the corresponding automaton as defined in Lemma 34. The output colours a, b, c, d have been replaced by 1, 2, 3, 4 for readability.

References

- 1 Bader Abu Radi and Orna Kupferman. Minimizing GFG transition-based automata. In *ICALP*, volume 132 of *LIPICs*, pages 100:1–100:16, 2019. doi:10.4230/LIPICs.ICALP.2019.100.
- 2 Bader Abu Radi and Orna Kupferman. Minimization and canonization of GFG transition-based automata. *Log. Methods Comput. Sci.*, 18(3), 2022. doi:10.46298/lmcs-18(3:16)2022.
- 3 Souheib Baair and Alexandre Duret-Lutz. Mechanizing the minimization of deterministic generalized Büchi automata. In *FORTE*, volume 8461 of *Lecture Notes in Computer Science*, pages 266–283, 2014. doi:10.1007/978-3-662-43613-4_17.
- 4 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *FSTTCS*, page 16, 2018. doi:10.4230/LIPICs.FSTTCS.2018.16.
- 5 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
- 6 Frantisek Blahoudek, Alexandre Duret-Lutz, and Jan Strejcek. Senator 2 can complement generalized Büchi automata via improved semi-determinization. In *CAV*, volume 12225 of *Lecture Notes in Computer Science*, pages 15–27, 2020. doi:10.1007/978-3-030-53291-8_2.
- 7 Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michał Skrzypczak. On succinctness and recognisability of alternating good-for-games automata. *CoRR*, abs/2002.07278, 2020. arXiv:2002.07278.
- 8 Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In *CONCUR*, volume 140, pages 19:1–19:16, 2019. doi:10.4230/LIPICs.CONCUR.2019.19.
- 9 Udi Boker and Karoliina Lehtinen. History determinism vs. good for gameness in quantitative automata. In *FSTTCS*, volume 213, pages 38:1–38:20, 2021. doi:10.4230/LIPICs.FSTTCS.2021.38.
- 10 Udi Boker and Karoliina Lehtinen. When a little nondeterminism goes a long way: An introduction to history-determinism. *ACM SIGLOG News*, 10(1):24–51, 2023. doi:10.1145/3584676.3584682.
- 11 Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenholve. Half-positional objectives recognized by deterministic Büchi automata. In *CONCUR*, volume 243, pages 20:1–20:18, 2022. doi:10.4230/LIPICs.CONCUR.2022.20.
- 12 J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. URL: <http://www.jstor.org/stable/1994916>.
- 13 J. Richard Büchi. On a decision method in restricted second order arithmetic. *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.
- 14 Antonio Casares. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. In *CSL*, volume 216, pages 12:1–12:17, 2022. doi:10.4230/LIPICs.CSL.2022.12.

- 15 Antonio Casares. *Structural properties of automata over infinite words and memory for games (Propriétés structurelles des automates sur les mots infinis et mémoire pour les jeux)*. PhD thesis, Université de Bordeaux, France, 2023. URL: <https://theses.hal.science/tel-04314678>.
- 16 Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. From Muller to Parity and Rabin Automata: Optimal Transformations Preserving (History) Determinism. *TheoretCS*, Volume 3, April 2024. doi:10.46298/theoretics.24.12.
- 17 Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. On the size of good-for-games Rabin automata and its link with the memory in Muller games. In *ICALP*, volume 229, pages 117:1–117:20, 2022. doi:10.4230/LIPIcs.ICALP.2022.117.
- 18 Antonio Casares and Corto Mascle. The complexity of simplifying ω -automata through the alternating cycle decomposition. *CoRR*, abs/2401.03811, 2024. arXiv:2401.03811.
- 19 Krishnendu Chatterjee, Wolfgang Dvorák, Monika Henzinger, and Veronika Loitzenbauer. Conditionally optimal algorithms for generalized Büchi games. In *MFCSS*, volume 58, pages 25:1–25:15, 2016. doi:10.4230/LIPICSS.MFCSS.2016.25.
- 20 Edmund M. Clarke, I. A. Draghicescu, and Robert P. Kurshan. A unified approach for showing language inclusion and equivalence between various types of omega-automata. *Inf. Process. Lett.*, 46(6):301–308, 1993. doi:10.1016/0020-0190(93)90069-L.
- 21 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP*, pages 139–150, 2009. doi:10.1007/978-3-642-02930-1_12.
- 22 Costas Courcoubetis, Moshe Y. Vardi, Pierre Wolper, and Mihalis Yannakakis. Memory-efficient algorithms for the verification of temporal properties. *Formal Methods Syst. Des.*, 1(2/3):275–288, 1992. doi:10.1007/BF00121128.
- 23 Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 - A framework for LTL and ω -automata manipulation. In *ATVA*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129, 2016. doi:10.1007/978-3-319-46520-3_8.
- 24 Rüdiger Ehlers. Minimising deterministic Büchi automata precisely using SAT solving. In *Theory and Applications of Satisfiability Testing - SAT*, volume 6175 of *Lecture Notes in Computer Science*, pages 326–332, 2010. doi:10.1007/978-3-642-14186-7_28.
- 25 Javier Esparza, Orna Kupferman, and Moshe Y. Vardi. Verification. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 1415–1456. European Mathematical Society Publishing House, Zürich, Switzerland, 2021. doi:10.4171/AUTOMATA-2/16.
- 26 Rob Gerth, Doron A. Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *IFIP*, volume 38, pages 3–18, 1995.
- 27 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer Science Logic*, pages 395–410, 2006. doi:10.1007/11874683_26.
- 28 John E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. Technical report, Stanford University, 1971. doi:10.5555/891883.
- 29 Christopher Hugenroth. Zielonka DAG acceptance, regular languages over infinite words. In *DLT*, 2023.
- 30 Sudeep Juvekar and Nir Piterman. Minimizing generalized Büchi automata. In *CAV*, pages 45–58, 2006.
- 31 Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In *ICALP*, pages 299–310, 2015. doi:10.1007/978-3-662-47666-6_24.
- 32 Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. Relating word and tree automata. In *LICS*, pages 322–332, 1996. doi:10.1109/LICS.1996.561360.
- 33 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω -pushdown automata. In *LICS*, page 689–702, 2020. doi:10.1145/3373718.3394737.
- 34 Thibaud Michaud and Maximilien Colange. Reactive synthesis from LTL specification with Spot. In *SYNT@CAV*, Electronic Proceedings in Theoretical Computer Science, 2018.
- 35 Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theor. Comput. Sci.*, 32:321–330, 1984. doi:10.1016/0304-3975(84)90049-5.

- 36 Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *SCT*, pages 157–168, 1984. doi:[10.1007/3-540-16066-3_15](https://doi.org/10.1007/3-540-16066-3_15).
- 37 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, page 179–190, 1989. doi:[10.1145/75277.75293](https://doi.org/10.1145/75277.75293).
- 38 Etienne Renault, Alexandre Duret-Lutz, Fabrice Kordon, and Denis Poitrenaud. Three SCC-based emptiness checks for generalized Büchi automata. In *LPAR*, volume 8312 of *Lecture Notes in Computer Science*, pages 668–682, 2013. doi:[10.1007/978-3-642-45221-5_44](https://doi.org/10.1007/978-3-642-45221-5_44).
- 39 Etienne Renault, Alexandre Duret-Lutz, Fabrice Kordon, and Denis Poitrenaud. Variations on parallel explicit emptiness checks for generalized Büchi automata. *Int. J. Softw. Tools Technol. Transf.*, 19(6):653–673, 2017. doi:[10.1007/S10009-016-0422-5](https://doi.org/10.1007/S10009-016-0422-5).
- 40 Schmuel Safra. On the complexity of ω -automata. In *FOCS*, page 319–327, 1988. doi:[10.1109/SFCS.1988.21948](https://doi.org/10.1109/SFCS.1988.21948).
- 41 Sven Schewe. Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete. In *FSTTCS*, volume 8, pages 400–411, 2010. doi:[10.4230/LIPIcs.FSTTCS.2010.400](https://doi.org/10.4230/LIPIcs.FSTTCS.2010.400).
- 42 Sven Schewe. Minimising Good-For-Games automata is NP-complete. In *FSTTCS*, volume 182, pages 56:1–56:13, 2020. doi:[10.4230/LIPIcs.FSTTCS.2020.56](https://doi.org/10.4230/LIPIcs.FSTTCS.2020.56).
- 43 Fabio Somenzi and Roderick Bloem. Efficient Büchi automata from LTL formulae. In *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263, 2000. doi:[10.1007/10722167_21](https://doi.org/10.1007/10722167_21).
- 44 Larry Stockmeyer. Planar 3-colorability is polynomial complete. *SIGACT News*, 5(3):19–25, jul 1973. doi:[10.1145/1008293.1008294](https://doi.org/10.1145/1008293.1008294).
- 45 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1 – 37, 1994. doi:[10.1006/inco.1994.1092](https://doi.org/10.1006/inco.1994.1092).

A G₂ conjecture

To establish that deciding whether a Büchi automaton is history-deterministic can be done in PTIME, Bagnol and Kuperberg [4] introduced the G_2 game, played on an automaton \mathcal{A} between two players Eve and Adam. Polynomial-time decidability is obtained using the facts that (1) the winner of this game can be decided in polynomial time, and, (2) for Büchi and coBüchi automata, \mathcal{A} is history-deterministic if and only if Eve wins G_2 [31, 4, 7]. We discuss now the status of this problem for generalised Büchi and generalised coBüchi automata.

Definitions and state of the art

We recall the G_2 game from [4]. Given an automaton \mathcal{A} , the G_2 game on \mathcal{A} , noted $G_2(\mathcal{A})$, is played between Eve and Adam in the following way: The arena is Q^3 , with starting position (q_0, q_0, q_0) . At each round, from position (p, q_1, q_2) :

1. Adam chooses a letter $a \in \Sigma$,
2. Eve chooses a transition $p \xrightarrow{a} p'$.
3. Adam chooses transitions $q_1 \xrightarrow{a} q'_1$ and $q_2 \xrightarrow{a} q'_2$.
4. The game moves to position (p', q'_1, q'_2)

Eve wins the game if either her run ρ on the first component is accepting, or both of Adam's runs λ_1 and λ_2 on second and third components are rejecting.

The G_2 conjecture, stated in [4] is the following:

► **Conjecture 35.** *A parity automaton \mathcal{A} is history-deterministic if and only if Eve wins $G_2(\mathcal{A})$.*

So far, the conjecture has been proved for Büchi automata [4, Cor. 21] and for coBüchi automata [7, Thm. 28].

A general remark: Composition by deterministic automata preserves G_2

We state here a property of the G_2 conjecture with respect to generic conditions. This idea has been present for some time among researchers in the field, we explicit it here for clarity.

► **Lemma 36.** *Let $W_1 \subseteq \Sigma^\omega$ and $W_2 \subseteq \Gamma^\omega$ be languages on finite alphabets Σ, Γ . Assume that the G_2 conjecture holds for W_2 -automata, and that there is a deterministic W_2 -automaton recognising W_1 . Then the G_2 conjecture holds for W_1 -automata.*

Proof. Let \mathcal{D} be the deterministic W_2 -automaton recognising W_1 . Let \mathcal{A} be an arbitrary non-deterministic W_1 -automaton. We want to prove that the G_2 conjecture holds for the automaton \mathcal{A} . We build a non-deterministic W_2 -automaton $\mathcal{B} = \mathcal{D} \circ \mathcal{A}$. Since \mathcal{D} is deterministic, the projection from runs of \mathcal{B} to runs of \mathcal{A} by forgetting the \mathcal{D} component is a bijection, that preserves acceptance. In particular $L(\mathcal{A}) = L(\mathcal{B})$.

Let us assume that Eve wins $G_2(\mathcal{A})$. By ignoring the \mathcal{D} component, Eve can use the same strategy to win $G_2(\mathcal{B})$. Since \mathcal{B} is a W_2 -automaton, by assumption we obtain that \mathcal{B} is HD. The HD strategy in \mathcal{B} can now be used in \mathcal{A} , using \mathcal{D} as extra memory. Indeed, Eve can win the letter game of \mathcal{A} by simulating the \mathcal{D} component in her strategy, and play as in the letter game of \mathcal{B} . We conclude that \mathcal{A} is HD, hence the G_2 conjecture holds for \mathcal{A} , as for any W_1 -automaton. ◀

Consequences: G_2 for generalised (co)Büchi automata

The first consequence that we can notice is that the G_2 conjecture on parity automata as stated in [4] suffices to imply the G_2 conjecture for all ω -automata. This follows from Lemma 36 and the fact that deterministic parity automata are sufficient to recognize any ω -regular language [36]. This feature of the G_2 conjecture was already mentioned e.g. in the survey [10, Sect 6.1].

We finally state another consequence of Lemma 36.

► **Theorem 37.** *The G_2 conjecture holds for generalised Büchi and generalised coBüchi automata.*

Proof. Both cases are a simple application of Lemma 36, as the G_2 conjecture holds for both Büchi [4, Cor. 21] and coBüchi automata [7, Thm. 28], and every generalised (co)Büchi condition is recognisable by a deterministic (co)Büchi automaton (see Section 3.3). ◀

Moreover, the G_2 game is still tractable for generalised Büchi and coBüchi conditions:

► **Lemma 38.** *The G_2 game for generalised Büchi and coBüchi automata can be solved in polynomial time.*

Proof. A GR(1) objective is an objective of the form $B_1 \Rightarrow B_2$, where B_1, B_2 are generalised Büchi objectives. It is known that GR(1) games can be solved in polynomial time [30, 19].

The objective of the G_2 game of a generalised coBüchi automaton is of the form $(C_1 \vee C_2) \Rightarrow C_3$, where C_1, C_2, C_3 are generalised coBüchi objectives. Since $C_1 \vee C_2$ is still a generalised coBüchi objective, taking the contrapositive yields a GR(1) objective $\neg C_3 \Rightarrow (\neg C_1 \wedge \neg C_2)$. Thus the G_2 game for a coBüchi automaton is a GR(1) game, and can be solved in PTIME.

For generalised Büchi automata, it is a little more complex, because the disjunction of two generalised Büchi objectives is not directly a generalised Büchi objective. However, it can be turned into one with a quadratic blow-up in the number of colours. Indeed, this corresponds to performing the following formula expansion:

$$\left(\bigwedge_{i \in I} A_i\right) \vee \left(\bigwedge_{j \in J} B_j\right) = \bigwedge_{i \in I} \bigwedge_{j \in J} A_i \vee B_j$$

Thus, we can turn the objective of a generalised Büchi G_2 game into a GR(1) objective, where the premise has quadratically many colours, with accepting sets of the form $A_i \cup B_j$. This means that as before, solving the G_2 game for a generalised Büchi automaton can be done in PTIME. ◀

This gives an alternative proof of Corollary 10.

B Correctness and minimality of $\mathcal{A}_L^{\text{genCoB}}$: Proofs for Section 4.3

We first prove Propositions 20 and 23.

For convenience, we let $\phi_i(q) = p_1$ for all $q \notin \mathcal{S}_i$ (so that $\phi_i: Q_{\mathcal{A}_L^{\text{CoB}}} \rightarrow Q$ is total).

► **Proposition 23 (Correctness).** *Let L be a coBüchi recognisable language. The automaton $\mathcal{A}_L^{\text{genCoB}}$ is history-deterministic and recognises L .*

Proof. Let $w \in \mathcal{L}(\mathcal{A}_L^{\text{genCoB}})$, and let $\rho \in \Delta^\omega$ be an accepting run over w . By definition of the generalised coBüchi acceptance condition, there is a suffix ρ' of ρ and some $i \in [k]$ such that i does not appear in the output of ρ' . By definition of the labelling col of $\mathcal{A}_L^{\text{genCoB}}$, this

means that ρ' is the ϕ_i -projection of a path in \mathcal{S}_i . Therefore, by Lemma 13, $w \in L$. This shows that $\mathcal{L}(\mathcal{A}_L^{\text{genCoB}}) \subseteq L$.

We now define a resolver $\sigma: \Sigma^* \times \Sigma \rightarrow \Delta$ for $\mathcal{A}_L^{\text{genCoB}}$ accepting any word from L , thereby completing the proof that $\mathcal{A}_L^{\text{genCoB}}$ is an HD automaton recognising L . The resolver will try to follow the different safe components of $\mathcal{A}_L^{\text{coB}}$ in a round-robin manner, by using k memory states. Let σ_0 be a resolver for $\mathcal{A}_L^{\text{coB}}$. We first remark that, by safe determinism of $\mathcal{A}_L^{\text{coB}}$, for all state q in $\mathcal{A}_L^{\text{genCoB}}$ and for all $a \in \Sigma$, there is at most one transition (q, a, q') in $\phi_i(\mathcal{S}_i)$, for each i . The resolver σ will use k memory states. Assume that we have read so far $u \in \Sigma^*$, reaching a state q , and letter $a \in \Sigma$ is given. If the resolver is in the i^{th} memory state, σ will indicate to take, if it exists, the only a -transition from q available in $\phi_i(\mathcal{S}_i)$. In this case, the memory state of σ is not updated. If, on the other hand, no such transition exists, then σ will choose the transition $(q, a, \phi_{i+1}(\sigma_0^*(ua)))$, and update its memory state to $i + 1$ (or 1, if $i = k$). That is, we look at the state reached in $\mathcal{A}_L^{\text{coB}}$ following the resolver σ_0 , and jump to its $i + 1$ -projection in $\mathcal{A}_L^{\text{genCoB}}$ and start simulating the run as if we where in the $(i + 1)^{\text{th}}$ safe component \mathcal{S}_{i+1} . We show that σ builds an accepting run whenever the input word is in L . Let $w \in L$, and let ρ be the accepting run over w built by the resolver σ_0 in $\mathcal{A}_L^{\text{coB}}$. Eventually, ρ will stay in a safe component \mathcal{S}_j . Let N be a large enough index such that the suffix of ρ after position N does not leave \mathcal{S}_j . Suppose by contradiction that the run on w built by σ in $\mathcal{A}_L^{\text{genCoB}}$ was rejecting. Then, it sees all output colours in $\{1, \dots, k\}$ infinitely often, therefore, it leaves each component $\phi_i(\mathcal{S}_i)$ infinitely many times. However, whenever this run enters $\phi_j(\mathcal{S}_j)$ after position N , it will coincide with the ϕ_j -projection of ρ' , so it will not leave this component, a contradiction. \blacktriangleleft

We now provide all details for the proof of minimality of $\mathcal{A}_L^{\text{genCoB}}$.

► **Proposition 24 (Minimality).** *Let \mathcal{B} be a history-deterministic generalised coBüchi automaton recognising a language L . Then, $|\mathcal{A}_L^{\text{genCoB}}| \leq |\mathcal{B}|$.*

Localisation at a residual. We recall that the local alphabet at a residual $R = u^{-1}L \in \text{Res}(L)$ is:

$$\Sigma|_R = \{v \in \Sigma^+ \mid [uv] = [u] \text{ and for any proper prefix } v' \text{ of } v, [uv'] \neq [v]\}.$$

We note that this definition is independent from the choice of the representative u .

We identify words over $\Sigma|_R$ with words over Σ . Note that, if it is non-empty, $\Sigma|_R$ is a prefix code, and therefore an infinite word $w \in \Sigma^\omega$ admits at most one decomposition of the form $w_1 w_2 \dots$, with $w_i \in \Sigma|_R$. As noted before, $\Sigma|_R$ is empty if and only if all the states of Q^R are transient, that is, they do not occur in any cycle of the automaton. If this is the case, we say that the residual R is *transient*, on the contrary, we say that it is *recurrent*.

For R a recurrent residual we let:

$$L|_R = \{w \in \Sigma|_R^\omega \mid w \in R\},$$

which is a prefix-independent language.

Also, for a recurrent residual R of L , the automaton $\mathcal{A}|_R$ is the generalised coBüchi automaton over $\Sigma|_R$ obtained by restricting \mathcal{A} to the states recognising R .

The following lemma directly follows from the definition.

► **Lemma 26.** *The automaton $(\mathcal{A}|_R)^q$ recognises $L|_R$ for each $q \in Q^R$. Moreover, if \mathcal{A}^q is history-deterministic, so is $(\mathcal{A}|_R)^q$.*

Safe components and safe languages of the localisation.

► **Lemma 39.** *Let \mathcal{A} be a semantically deterministic coBüchi automaton. Two states $q, p \in Q^R$ are in a same safe component in \mathcal{A} if and only if they are in a same safe component in $\mathcal{A}|_R$. Also, for all $q \in Q^R$, the safe language of q in $\mathcal{A}|_R$ is $\mathcal{L}_{\text{Safe}}(\mathcal{A}^q) \cap \Sigma|_R^\omega$.*

Proof. Follows from the fact that (safe) paths between states in Q^R are the same in \mathcal{A} or in $\mathcal{A}|_R$ (note that here $\mathcal{A}|_R$ is a coBüchi automaton). ◀

Lemma 27 follows from the following Lemma, combined with Theorem 16.

► **Lemma 40.** *If R is a recurrent residual of L , then $\mathcal{A}_L^{\text{coB}}|_R$ is nice, safe minimal and safe centralised. If R is transient, then $\mathcal{A}_L^{\text{coB}}$ contains a single state recognising R .*

Proof. Recall that by Theorem 17, $\mathcal{A}_L^{\text{coB}}$ is nice, safe minimal and safe centralised.

First, assume that R is transient, and let q be a state of $\mathcal{A}_L^{\text{coB}}$ recognising R . Since $\mathcal{A}_L^{\text{coB}}$ is in normal form and all transitions outgoing from q change of strongly connected component, the safe language of q is the empty set. If there is $p \sim q$, $p \neq q$, the state p would appear in a different safe component of $\mathcal{A}_L^{\text{coB}}$, contradicting the fact that it is safe centralised.

Assume that R is recurrent. It is a direct check that $\mathcal{A}_L^{\text{coB}}|_R$ is nice. Safe minimality and safe centrality follow from Lemma 39. ◀

We can finally combine previous lemmas together with Proposition 21 to obtain minimality of $\mathcal{A}_L^{\text{genCoB}}$ in the general case.

Proof of Proposition 24. Let \mathcal{B} be an HD generalised coBüchi automaton recognising L . We use the notations introduced at the beginning of this subsection: $\text{Res}(L) = \{R_1, \dots, R_m\}$, and n_j is the maximal number of states corresponding to R_j appearing in a same safe component of $\mathcal{A}_L^{\text{coB}}$.

We claim that $|Q_{\mathcal{B}}^{R_j}| \geq n_j$. This will conclude the proof, as it implies:

$$|\mathcal{B}| \geq n_1 + \dots + n_m = |\mathcal{A}_L^{\text{genCoB}}|.$$

First, if R_j is a transient residual, by Lemma 27, $n_j = 1$, so the inequality holds.

Assume that R_j is recurrent. By Lemmas 26 and 27, $\mathcal{A}_L^{\text{coB}}|_{R_j}$ is a minimal HD coBüchi automaton recognising $L|_{R_j}$ (which is a prefix-independent language). Moreover, by Lemma 39, $\mathcal{A}_L^{\text{coB}}|_{R_j}$ contains a safe component of size n_j . Therefore, by Proposition 21, an HD generalised coBüchi automaton recognising $L|_{R_j}$ has at least n_j states. We have, by Lemma 26, that $\mathcal{B}|_{R_j}$ is an HD generalised coBüchi automaton recognising $L|_{R_j}$, so we conclude that $|Q_{\mathcal{B}}^{R_j}| = |\mathcal{B}|_{R_j}| \geq n_j$, as we wanted. ◀

C NP-completeness of minimisation for (history-)deterministic generalised Büchi automata: Proofs for Section 5

C.1 Upper bound: Proof of Proposition 41

► **Proposition 41.** *The minimisation of deterministic generalised Büchi and generalised coBüchi and history-deterministic generalised Büchi automata are in NP.*

We establish the NP-upper bounds for state minimisation of deterministic and history-deterministic generalised Büchi automata. The result for deterministic generalised coBüchi automata follows by duality (Remark 5).

We start by establishing our key technical lemma, from which the proposition will follow. It states that the number of colours needed by a history-deterministic Büchi automaton is polynomial in its size and the size of a minimal equivalent deterministic Büchi automaton.

We say that we can *recolour* a generalised (co)Büchi automaton \mathcal{A} with k colours if we can replace its colour labelling with one using k colours without changing the language.

A history-deterministic automaton is called *resolver-trim* if there is a resolver σ for it such that for every state q there is an accepting run induced by σ that goes through q . Note that the notions of resolver-trim matches the notion of trim over deterministic automata.

► **Lemma 42.** *Let \mathcal{A} be a resolver-trim history-deterministic generalised Büchi automaton with $n_{\mathcal{A}}$ states. If there exists an equivalent deterministic Büchi automaton \mathcal{B} with $n_{\mathcal{B}}$ states, then one can recolour \mathcal{A} to obtain an equivalent resolver-trim history-deterministic generalised Büchi automaton with $n_{\mathcal{A}}$ states and using at most $n_{\mathcal{A}}n_{\mathcal{B}}$ colours.*

Proof. Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, q_{\text{init}}^{\mathcal{A}}, \Delta_{\mathcal{A}}, \Gamma_{\mathcal{A}}, \text{col}_{\mathcal{A}}, \text{genB}_{\Gamma})$ be a resolver-trim history-deterministic generalised Büchi automaton with $n_{\mathcal{A}} = |Q_{\mathcal{A}}|$ states and using $k = |\Gamma|$ colours. Let σ be a resolver for \mathcal{A} such that every state is visited by an accepting run following σ . Let $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, q_{\text{init}}^{\mathcal{B}}, \Delta_{\mathcal{B}}, \{1\}, \text{col}_{\mathcal{B}}, \text{genB}_{\{1\}})$ be a deterministic Büchi automaton with $n_{\mathcal{B}} = |Q_{\mathcal{B}}|$ states such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

We prove the lemma by showing that either we can remove a colour from \mathcal{A} or $k \leq n_{\mathcal{A}}n_{\mathcal{B}}$. For all $i \in \Gamma_{\mathcal{A}}$ let \mathcal{A}_{-i} be the automaton obtained by removing the colour i from \mathcal{A} , i.e., composing $\text{col}_{\mathcal{A}}$ with a projection over $\Gamma_{\mathcal{A}} \setminus \{i\}$ and replacing the acceptance condition by $\text{genB}_{\Gamma \setminus \{i\}}$. Note that every accepting run in \mathcal{A} is still accepting in \mathcal{A}_{-i} . Hence we must have $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}_{-i})$.

- First suppose there exists i such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{-i})$. As every accepting run in \mathcal{A} is still accepting in \mathcal{A}_{-i} , σ is still a resolver for \mathcal{A}_{-i} . As a result, \mathcal{A}_{-i} is a resolver-trim history-deterministic generalised Büchi automaton with the same states and transitions as \mathcal{A} and using $k - 1$ colours.
- Now suppose that for all $i \in \Gamma_{\mathcal{A}}$ we have $\mathcal{L}(\mathcal{A}) \subsetneq \mathcal{L}(\mathcal{A}_{-i})$. As $\mathcal{L}(\mathcal{A}_{-i}) \setminus \mathcal{L}(\mathcal{A})$ is ω -regular and non-empty, it contains an ultimately periodic word, which itself has an ultimately periodic accepting run in \mathcal{A}_{-i} . As a consequence, we can find a state $q_i^{\mathcal{A}}$ and words u_i, v_i , such that $u_i v_i^{\omega}$ is not accepted by \mathcal{A} and there exists a path $q_{\text{init}}^{\mathcal{A}} \xrightarrow{u_i} q_i^{\mathcal{A}}$ and a cycle $q_i^{\mathcal{A}} \xrightarrow{v_i} q_i^{\mathcal{A}}$ whose output contains all colours of $\Gamma_{\mathcal{A}}$ except i .

We show that $k \leq n_{\mathcal{A}}n_{\mathcal{B}}$. Suppose by contradiction that $k > n_{\mathcal{A}}n_{\mathcal{B}}$. By the pigeonhole principle, there exists $q^{\mathcal{A}} \in Q_{\mathcal{A}}$ such that at least $n_{\mathcal{B}} + 1$ distinct colours $i \in \Gamma_{\mathcal{A}}$ satisfy $q_i^{\mathcal{A}} = q^{\mathcal{A}}$. Let $I = \{i \in \Gamma_{\mathcal{A}} \mid q_i^{\mathcal{A}} = q^{\mathcal{A}}\}$.

By definition of σ , there is a word $u \in \Sigma^*$ such that the path induced by σ when reading u from $q_{\text{init}}^{\mathcal{A}}$ ends in $q^{\mathcal{A}}$. For all $i \in I$, $u v_i^{\omega}$ cannot be in $\mathcal{L}(\mathcal{A})$, as otherwise \mathcal{A} would contain an accepting run over v_i^{ω} from $q_{\mathcal{A}}$, meaning that $u_i v_i^{\omega}$ would also be in $\mathcal{L}(\mathcal{A})$, a contradiction. Hence $u v_i^{\omega}$ is not in $\mathcal{L}(\mathcal{A})$ for any $i \in I$.

Let $q^{\mathcal{B}}$ be the state reached in \mathcal{B} by reading u from $q_{\text{init}}^{\mathcal{B}}$. For each $i \in I$, we can find a state $q_i^{\mathcal{B}}$ in \mathcal{B} and $\alpha_i, \beta_i > 0$ such that there is a path $q^{\mathcal{B}} \xrightarrow{v_i^{\alpha_i}} q_i^{\mathcal{B}}$ and a cycle $q_i^{\mathcal{B}} \xrightarrow{v_i^{\beta_i}} q_i^{\mathcal{B}}$. Furthermore, as $u v_i^{\omega}$ is not accepted by \mathcal{A} , it is not accepted by \mathcal{B} and thus there are no Büchi transitions along that cycle.

By the pigeonhole principle, there exist $i \neq j$ such that $q_i^{\mathcal{B}} = q_j^{\mathcal{B}}$. As a result, the word $u v_i^{\alpha_i} (v_i^{\beta_i} v_j^{\beta_j})^{\omega}$ is not accepted by \mathcal{B} : its run in \mathcal{B} goes to $q^{\mathcal{B}}$ and then goes indefinitely through the cycles reading $v_i^{\beta_i}$ and $v_j^{\beta_j}$, which, as we saw, contain no Büchi transition.

However, $uv_i^{\alpha_i}(v_i^{\beta_i}v_j^{\beta_j})^\omega$ has an accepting run in \mathcal{A} : we can read u to arrive in $q_{\mathcal{A}}$, and then read every factor v_i (resp. v_j) by going from $q_{\mathcal{A}}$ to itself and seeing every colour but i (resp. j). As $i \neq j$, this run sees every colour infinitely many times. This is a contradiction, as we assumed $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. ◀

This lemma allows us to recolour the minimal automata with a polynomial number of colours in the input.

► **Lemma 31.** *Let \mathcal{A} be a deterministic generalised Büchi automaton with n states and k colours. Then there is an equivalent deterministic generalised Büchi automaton with a minimal number of states and using $O(n^2k)$ colours.*

Proof. By Corollary 9 there exists a deterministic Büchi automaton \mathcal{A}' equivalent to \mathcal{A} and with nk states. Let \mathcal{B} be a minimal deterministic generalised Büchi automaton equivalent to \mathcal{A} . By minimality, \mathcal{B} has at most n states, hence by Lemma 42 we can recolour it using at most n^2k colours. As recolouring preserves determinism, we obtain the lemma. ◀

► **Lemma 32.** *Let \mathcal{A} be a history-deterministic generalised Büchi automaton with n states and k colours. Then there is an equivalent history-deterministic generalised Büchi automaton with a minimal number of states and using $O(n^3k^2)$ colours.*

Proof. By Corollary 9 there exists a history-deterministic Büchi automaton \mathcal{A}' equivalent to \mathcal{A} and with nk states. Using the quadratic determinisation construction for history-deterministic Büchi automata [31, Theorem 8], we can construct a deterministic Büchi automaton \mathcal{A}'' equivalent to \mathcal{A} and with at most $O(n^2k^2)$ states. Let \mathcal{B} be a minimal history-deterministic generalised Büchi automaton equivalent to \mathcal{A} . By minimality, \mathcal{B} has at most n states and is resolver-trim, hence by Lemma 42 we can recolour it using at most $O(n^3k^2)$ colours, yielding the lemma. ◀

We can now conclude the proof of Proposition 41.

Proof of Proposition 41. We consider the cases of deterministic and history-deterministic automata separately:

For deterministic automata: Given a bound n and a deterministic generalised Büchi automaton \mathcal{A} with $n_{\mathcal{A}}$ states and $k_{\mathcal{A}}$ colours, one can guess a deterministic generalised Büchi automaton \mathcal{B} with at most n states and $O(n_{\mathcal{A}}^2k_{\mathcal{A}})$ colours. It suffices then to check that \mathcal{A} and \mathcal{B} are equivalent. This can be done in polynomial time by turning both automata into equivalent deterministic Büchi ones (Corollary 9) and then checking equivalence between the resulting deterministic Büchi automata, which can be done in polynomial time [20].

For history-deterministic automata: Given a bound n and a HD generalised Büchi automaton \mathcal{A} with $n_{\mathcal{A}}$ states and $k_{\mathcal{A}}$ colours, one can guess an HD generalised Büchi automaton \mathcal{B} with at most n states and $O(n_{\mathcal{A}}^3k_{\mathcal{A}}^2)$ colours. It suffices then to check that \mathcal{A} and \mathcal{B} are equivalent. This can be done in polynomial time by turning both automata into equivalent HD Büchi ones (Corollary 9) and then checking equivalence between the resulting HD Büchi automata, which can be done in polynomial time via a simulation game, see for instance [42, Thm. 4]⁴. ◀

⁴ Schewe's proof concerns state-based automata, but the exact same proof works for transition-based ones. One can also use his work as a black box by translating both automata into equivalent state-based ones (with an at most quadratic blow-up) and then using Schewe's method to check equivalence.

We then prove that the number of colours may increase exponentially when minimising a non-deterministic generalised Büchi automaton. A symmetric proof shows that it is also the case for non-deterministic generalised coBüchi automata.

► **Proposition 33.** *There exists a family of non-deterministic generalised Büchi automata $(\mathcal{A}_n)_{n \in \mathbb{N}}$ such that for all n , \mathcal{A}_n uses $n + 1$ states and 2 colours and a minimal automaton equivalent to \mathcal{A}_n requires 2^n colours.*

Proof. We define \mathcal{A}_n as follows: Its set of states is $Q = \{q_{\text{init}}, q_1, \dots, q_n\}$, with q_{init} the initial state, its input alphabet is $\Sigma = \{1, \dots, 2n\}$, and its output alphabet is $\Gamma = \{\alpha, \beta\}$. For all $i, j \in \{1, \dots, n\}$ there are transitions $q_{\text{init}} \xrightarrow{i:\varepsilon} q_j$, $q_i \xrightarrow{2i-1:\alpha} q_i$ and $q_i \xrightarrow{2i:\beta} q_i$. There are no other transitions. This automaton recognises the language

$$\{w \in \Sigma^\omega \mid \exists i \in \{1, \dots, n\}, \{2i - 1, 2i\} \subseteq \text{Inf}(w)\}.$$

It can also be described as the set of words w such that $\text{Inf}(w)$ satisfies $\phi = (1 \wedge 2) \vee \dots \vee (2n - 1 \wedge 2n)$, where each letter is interpreted as \top if it is in the set and \perp otherwise. This DNF formula can be put in CNF, but the resulting formula has exponentially more clauses:

$$\bigwedge_{i_1 \in \{1, 2\}} \dots \bigwedge_{i_n \in \{2n-1, 2n\}} (i_1 \vee \dots \vee i_n).$$

From this observation we can build an equivalent (deterministic) generalised Büchi automaton with a single state: it suffices to assign a colour to each of those clauses, and for each letter $i \in \Sigma$ put a i -loop labelled by the colours corresponding to clauses containing i .

Moreover we cannot use less than 2^n colours for an equivalent automaton with one state: Suppose there exists such an automaton, let Γ be its set of output colours. For each $\gamma \in \Gamma$ let Σ_γ be the set of letters a such that γ appears in a loop reading a . The language of that automaton is then the set of words w such that $\text{Inf}(w)$ satisfies $\bigwedge_{\gamma \in \Gamma} \bigvee_{i \in \Sigma_\gamma} i$. If $|\Gamma| < 2^n$, then we have a CNF formula equivalent to ϕ with less than 2^n clauses. It is folklore that such a formula does not exist, but we reprove it here for completeness.

Let ψ be a CNF formula over Σ equivalent to ϕ . Let $V \subseteq 2^\Sigma$ be the set of sets of colours which contain exactly one of $2i - 1$ or $2i$ for each $i \in \{1, \dots, n\}$. Each clause in ψ must contain either $2i - 1$ or $2i$ for each $i \in \{1, \dots, n\}$, as otherwise $\{2i - 1, 2i\}$ would not satisfy ψ . As a consequence, each clause is satisfied by all sets of V except at most one. As ψ is equivalent to ϕ , which is satisfied by no set of V , ψ must have at least 2^n clauses: each one allowing to rule out one of the sets of V .

In conclusion, every one-state automaton equivalent to \mathcal{A}_n must use at least 2^n colours. ◀

C.2 NP-completeness of state minimisation: Proof of Theorems 28 and 29

We start by presenting a version of the 3-colouring that will be used to show NP-hardness of the minimisation of (history-)deterministic Büchi automata.

Colourings of triangle-full 4-clique-free graph

A *finite undirected graph* is a pair $G = (V, E)$ with V a finite set of *vertices* and $E \subseteq \binom{V}{2}$ a set of pairs of vertices, called *edges*. In all that follows we will simply use the term *graph* for a finite undirected graph. We say that two vertices $u, v \in V$ are *neighbours* if $\{u, v\} \in E$.

Recall that we defined the *neighbourhood* of a vertex v as the set $N[v] = \{v' \in V \mid \{v, v'\} \in E\}$, and its *strict neighbourhood* as the set $n(v) = N[v] \setminus \{v\}$. Let $k \in \mathbb{N}$, a k -*clique* is a set of k vertices $\{v_1, \dots, v_k\}$ such that $\{v_i, v_j\} \in E$ for all $i \neq j \in \{1, \dots, k\}$.

We say that a graph is *triangle-full* if all vertices are in a 3-clique. It is *4-clique-free* if it does not have any 4-clique.

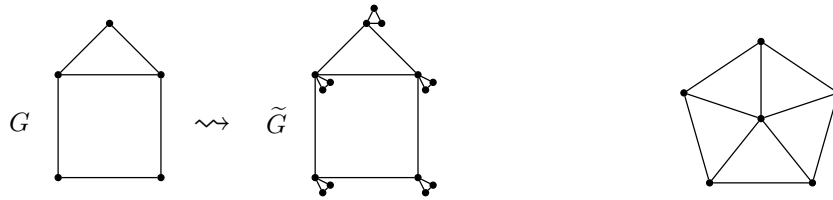
A k -*colouring* of a graph G is a function $c : V \rightarrow \{1, \dots, k\}$ such that for all $\{v, v'\} \in E$, $c(v) \neq c(v')$. A graph is k -*colourable* if there exists a k -colouring of it. The *colouring problem* asks, given a graph and an integer k , if that graph is k -colourable. The 3-colouring problem asks if a given graph is 3-colourable. Those problems are well known to be NP-complete [44]. They are in fact already NP-complete for triangle-full 4-clique-free graphs:

► **Lemma 43.** *The following problem is NP-hard: Given a triangle-full 4-clique-free graph, is it 3-colourable?*

Proof. We reduce from the general 3-colouring problem. Let $G = (V, E)$ be a graph. We define a triangle-full 4-clique-free graph \tilde{G} such that G is 3-colourable if and only if \tilde{G} is.

If G contains a 4-clique then it is not 3-colourable. We can check whether this is the case in polynomial time, and if so, we can simply set \tilde{G} as an arbitrary triangle-full 4-clique-free non-3-colourable graph, for instance the one in Figure 4.

Otherwise we define the graph $\tilde{G} = (V \times \{0, 1, 2\}, E_0 \cup E_{012})$, such that $E_0 = \{\{(v, 0), (v', 0)\} \mid \{v, v'\} \in E\}$ and $E_{012} = \{\{(v, i), (v, j)\} \mid v \in V, i \neq j \in \{0, 1, 2\}\}$.



■ **Figure 4** On the left, an example of the reduction of Lemma 43. On the right, a triangle-full 4-clique-free graph that is not 3-colourable.

Clearly \tilde{G} is triangle-full, as $(v, 0), (v, 1), (v, 2)$ are connected to each other for all v . Furthermore, it does not contain a 4-clique as G does not and all additional vertices have only two neighbours.

Further, if \tilde{G} is 3-colourable, then so is G : it suffices to colour each vertex v in G with the colour of $(v, 0)$ in \tilde{G} . Conversely, if G is 3-colourable, then we can colour every vertex $(v, 0)$ in \tilde{G} with the colour of v in G , and then colour $(v, 1)$ and $(v, 2)$ with the two remaining colours.

As a result, G is 3-colourable if and only if \tilde{G} is. ◀

The reduction: Automata and graph colourings

We now associate with each triangle-full, 4-clique-free graph $G = (V, E)$ a language L_G such that L_G is recognised by a 3-state generalised Büchi automaton if and only if G is 3-colourable.

For each $v \in V$, we define

$$L_v = (V^*vv)^\omega \cup (V^*(V \setminus N[v]))^\omega \quad \text{and we let} \quad L_G = \bigcap_{v \in V} L_v.$$

In other words, a word over V is in L_G if for all $v \in V$ it either has infinitely many factors vv or sees a vertex that is not a neighbour of v infinitely many times.

We first prove that, given a graph G , we can build in polynomial time a deterministic generalised Büchi automaton recognising L_G .

► **Lemma 44.** *Given a graph $G = (V, E)$, we can build in polynomial time a deterministic generalised Büchi automaton recognising L_G .*

Proof. We define a generalised Büchi automaton over the alphabet $\Sigma = V$. Automaton \mathcal{A} is given by:

- The set of states is $Q = V$, we pick any state as the initial one.
- For $q \in Q$ and $v \in \Sigma = V$, the v -transition from q is $q \xrightarrow{v} v$.
- The set of output colours is V , hence the output alphabet is $\Gamma = 2^V$.
- If $q \neq v$, the transition $q \xrightarrow{v} v$ is coloured with $V \setminus N[v]$. Transitions of the form $v \xrightarrow{v} v$ are coloured with $V \setminus n(v)$.

That is, the automaton uses the structure of the graph; when reading a vertex v we jump to the state corresponding to v , and produce as output the colours corresponding to vertices that are not on its neighbourhood. Self-loops do moreover produce colour v .

Therefore, each time that either a vertex not in $N[v]$ or the factor vv is read, colour v is produced, and these are the only cases in which colour v is produced. Therefore, $\mathcal{L}(\mathcal{A}) = L_G$. ◀

We then extend the previous lemma: from a k -colouring of the graph we can construct an automaton with k states recognising L_G .

► **Lemma 34.** *For all graph $G = (V, E)$ and $k \in \mathbb{N}$, if G is k -colourable then there exists a complete deterministic generalised Büchi automaton \mathcal{B} with k states which recognises L_G .*

Proof. Suppose G is k -colourable, let $c : V \rightarrow \{1, \dots, k\}$ be a k -colouring of G . We define the deterministic generalised Büchi automaton \mathcal{B} as follows:

- The set of states is $Q = \{1, \dots, k\}$.
- For $q \in Q$ and $v \in \Sigma = V$, the v -transition from q is $q \xrightarrow{v} c(v)$.
- The set of output colours is V , hence the output alphabet is $\Gamma = 2^V$.
- If $q \neq v$, the transition $q \xrightarrow{v} v$ is coloured with $V \setminus N[v]$. Transitions of the form $v \xrightarrow{v} v$ are coloured with $V \setminus n(v)$.

We argue that $\mathcal{L}(\mathcal{B}) = L_G$. Let $w = v_0v_1 \dots \in L_G$, as \mathcal{B} is deterministic and complete w has a unique run in \mathcal{B} .

Let $v \in V$, w is in L_v . We show that the run of \mathcal{B} over w must see colour v infinitely often. We distinguish two cases:

- If vv appears infinitely often as a factor of w , then by definition of \mathcal{B} , after reading the first v the run reaches state $c(v)$. As a result, the transition reading the second v must be $c(v) \xrightarrow{v} c(v)$, which is labelled by $V \setminus n(v)$. In particular the colour v is seen infinitely often.
- Otherwise, there is a vertex $u \notin N[v]$ that is read infinitely often. As $u \notin N[v]$, $v \notin N[u]$, thus v appears in the label of all transitions reading u . As a result, the colour v is seen infinitely often.

We have shown that the run of \mathcal{B} over w sees each label of Γ_G infinitely often, hence the run is accepting. As a consequence, $L_G \subseteq \mathcal{L}(\mathcal{B})$.

For the other inclusion, let $w = v_0v_1 \dots \in \mathcal{L}(\mathcal{B})$, and let ρ be an accepting run of w in \mathcal{B} . For each $v \in V$, the label v is seen infinitely often in that run. We show that w must then

be in L_v . The transitions labelled by v are the ones reading vertices that are not neighbours of v and the loop on $c(v)$ reading v . We distinguish two cases:

- If w contains infinitely many occurrences of vertices of $V \setminus N[v]$ then it is in L_v .
- Otherwise, w ultimately only contains vertices of $N[v]$. As colour v is seen infinitely often in ρ , the loop $c(v) \xrightarrow{v} c(v)$ must be taken infinitely often. Furthermore, the transitions going to $c(v)$ are all labelled by vertices mapped to $c(v)$ by c . As c is a k -colouring of G , apart from v none of those vertices can be in $N[v]$. Therefore, eventually every execution of the loop $c(v) \xrightarrow{v} c(v)$ must be preceded by a transition reading v , which implies that the factor vv appears infinitely many times in w , and thus that w is in L_v .

We showed that w is in L_v for all v , hence $\mathcal{L}(\mathcal{B}) \subseteq L_G$. We have proven that \mathcal{B} recognises L_G , which concludes our proof. ◀

We must now show the other implication, that an automaton with 3 states recognising L_G induces a 3-colouring of G . To do so, we will rely on the notions of v -cycle and $n(v)$ -cycle.

Let $G = (V, E)$ be a graph, and \mathcal{A} an automaton over the alphabet V . A v -cycle (resp. $n(v)$ -cycle) C in \mathcal{A} is a non-empty set of states such that for all $q, q' \in C$, there is a non-empty path from q to q' with only transitions reading v (resp. letters in $n(v)$).

► **Lemma 45.** *For all triangle-full 4-clique-free graph $G = (V, E)$, if there exists a (non-deterministic) complete generalised Büchi automaton \mathcal{B} with three states recognising L_G then G is 3-colourable.*

Proof. We will proceed in two main steps: First we will show that for all $v \in V$ there is exactly one state q_v that is part of a v -cycle (Claim 45.2). Then we will show that two neighbours u, v cannot share that state, i.e., $q_u \neq q_v$ (Claim 45.3). As \mathcal{B} has three states, mapping each vertex v to q_v yields a 3-colouring of G .

At many points we will use the fact that as \mathcal{B} is a generalised Büchi automaton in the following way. If we have an accepting run ρ and a run ρ' such that every transition occurring infinitely often in ρ also occurs infinitely often in ρ' , then ρ' is accepting: as ρ sees all colours infinitely many times, so does ρ' .

Let us start with the following observation.

▷ **Claim 45.1.** For all $v \in V$, there exists $q \in Q$ such that q is not in a $n(v)$ -cycle.

Proof. Suppose by contradiction that each $q \in Q$ is in a $n(v)$ -cycle. Then there exists a word $w_q \in n(v)^+$ and a path from q to q reading w_q . Let $\{u_1, \dots, u_k\} = n(v)$. The word $(u_1^2 \cdots u_k^2 v^2)^\omega$ is in L_G , as for all $u \in V$ either this word contains infinitely many uu or u is not a neighbour of v , which appears infinitely many times. Consider an accepting run ρ over this word. We transform ρ by inserting after each v a path from q to itself reading w_q , where q is the state reached after reading that v .

We obtain a run reading a word in which all vertices read are in the neighbourhood of v , but which contains no vv factor, hence is not in L_G . However, all edges seen infinitely often in the first run are also seen infinitely often in the second one, thus the latter is accepting, a contradiction. ◀

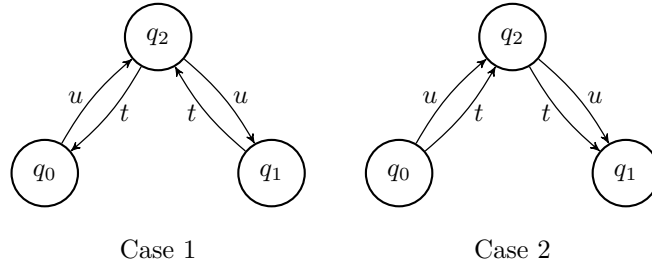
This first claim allows us to proceed with the first step of the proof: We argue in the following claim that for all vertex v , at most one state is in a v -cycle.

▷ **Claim 45.2.** For all $v \in V$ there is one and only one state $q_v \in Q_{\mathcal{B}}$ that is part of a v -cycle.

Proof. As we assumed that \mathcal{B} is complete, there must exist a v -cycle for all v (by reading v 's from any state we eventually encounter a cycle). The difficulty is then to show that two distinct states cannot both be in v -cycles.

Let $\{q_0, q_1, q_2\} = Q_{\mathcal{B}}$. Suppose by contradiction that there exists $v \in V$ such that two distinct states are in v -cycles. Without loss of generality we assume that it is the case for q_0 and q_1 . As G is triangle-full, v has two neighbours t and u such that $\{t, u\} \in E$. Furthermore, as G is 4-clique-free, t, u and v have no common neighbour, hence the word $(t^2u^2v^2)^\omega$ is in L_G . Let ρ be an accepting run reading that word in \mathcal{B} . As q_0 and q_1 are in v -cycles, for all $i \in \{0, 1\}$ there exists $k_i > 0$ and a path π_i from q_i to itself reading v^{k_i} . We construct a run ρ' by inserting in ρ , at every occurrence of q_0 and q_1 , the paths π_0 and π_1 , respectively. As ρ is accepting and all transitions taken infinitely often in ρ also appear infinitely often in ρ' , ρ' is accepting. Hence it must read a word of $\mathcal{L}(\mathcal{B}) = L_G$. As it only reads vertices t, u and v , in particular it must read t^2 and u^2 infinitely often. By construction of ρ' , the state reached between two consecutive t 's or u 's must be q_2 , as otherwise a path reading vs would have been inserted.

However, there cannot be a loop on q_2 reading t , or two transitions reading t to and from q_i for some $i \in \{0, 1\}$, as otherwise q_2 would be in a t -cycle, and thus all three states would be in a $n(u)$ -cycle, contradicting Claim 45.1. The only possibility is that for some $i \in \{0, 1\}$, there is a transition $q_i \xrightarrow{t} q_2$ and another one $q_2 \xrightarrow{t} q_{1-i}$, and no other transition reading t to or from q_2 . By the same argument, for some $j \in \{0, 1\}$, there is a transition $q_j \xrightarrow{u} q_2$ and another one $q_2 \xrightarrow{u} q_{1-j}$, and no other transition reading u to or from q_2 .



■ **Figure 5** An illustration of the case distinction from the proof of Claim 45.2

Case 1: $i \neq j$. Then the transitions reading t and u induce a $n(v)$ -cycle covering all states, contradicting Claim 45.1.

Case 2: $i = j$. Then we obtain a run ρ'' by applying the following transformation to ρ' . We showed that two consecutive t read in ρ' could only be read through the transitions $q_i \xrightarrow{t} q_2 \xrightarrow{t} q_{1-i}$. we replace each occurrence of those transitions with $q_i \xrightarrow{t} q_2 \xrightarrow{u} q_{1-i}$, which we can do as $i = j$. Similarly, we replace each occurrence of $q_i \xrightarrow{u} q_2 \xrightarrow{u} q_{1-i}$ with $q_i \xrightarrow{u} q_2 \xrightarrow{t} q_{1-i}$.

As both tt and uu are read infinitely often in ρ' , and the runs ρ' and ρ'' see the same set of transitions infinitely often, hence ρ'' is accepting. However, it is easy to see that ρ'' never reads two consecutive u but only reads vertices from $\{t, u, v\} \subseteq N[u]$, thus ρ'' is an accepting run of \mathcal{B} reading a word that is not in L_G , a contradiction.

We reached a contradiction in both cases, proving the claim. ◁

By Claim 45.2, we can define a mapping $c : V \rightarrow Q_{\mathcal{B}}$ such that for all $v \in V$, $\{c(v)\}$ is the only v -cycle. It remains to show that c is a 3-colouring.

▷ **Claim 45.3.** For all $(u, v) \in E$, $c(u) \neq c(v)$.

Proof. Suppose by contradiction that $c(u) = c(v)$. Let $\{u_1, \dots, u_k\} = n(v) \setminus \{u\}$. The word $(u^2 v^2 u_1^2 \dots u_k^2)^\omega$ is in L_G , and thus there is an accepting run ρ reading it. As $\{c(u)\}$ is the only u -cycle in \mathcal{B} , after reading u^2 the reached state can only be $c(u)$ (as \mathcal{B} has three states). Moreover, by reading v from $c(u) = c(v)$ we stay in $c(v)$, as $\{c(v)\}$ is the only v -cycle.

Hence after each factor $u^2 v$, the run ρ reaches $c(v)$. We construct ρ' by inserting additional transitions $c(v) \xrightarrow{u} c(v)$ at each occurrence of $c(v)$ in ρ . As ρ is accepting and the set of transitions occurring infinitely often in ρ' contains the ones of ρ , ρ' is accepting. However, ρ' reads only letters from $N[v]$, and never reads two consecutive v 's, thus it does not read a word of L_G , a contradiction. \triangleleft

By Claim 45.3, the map c is a 3-colouring of G , hence G is 3-colourable, proving the lemma. \blacktriangleleft

We can now conclude the proofs of Theorems 28 and 29.

► **Theorem 28.** *The following problem is NP-complete: Given a history-deterministic generalised Büchi automaton \mathcal{A} and a number n , decide whether there is an equivalent history-deterministic generalised Büchi automaton with at most n states.*

Proof. The upper bound follows from Proposition 41. For the lower bound, we have shown that the 3-colouring problem is NP-hard for triangle-full 4-clique-free graphs. Furthermore, given a triangle-full 4-clique-free graph G , we can construct in polynomial time an automaton \mathcal{A}_G recognising a language L_G (Lemma 44) such that:

- If G is 3-colourable then there is a deterministic (hence also history-deterministic) generalised Büchi automaton \mathcal{B} with three states recognising L_G (Lemma 34).
- If there is an generalised Büchi automaton \mathcal{B} with three states recognising L_G (in particular, if there is a history-deterministic one), then G is 3-colourable (Lemma 45).

Hence the 3-colouring problem over triangle-full 4-clique-free graphs reduces to the minimisation problem for HD generalised Büchi automata, which is therefore NP-hard. \blacktriangleleft

► **Theorem 29.** *The minimisation of the number of states of deterministic generalised Büchi and generalised coBüchi automata is NP-complete.*

Proof. We only need to prove the generalised Büchi case, as NP-completeness for generalised coBüchi automata follows by duality (Remark 5). We proceed similarly as in the previous proof.

The upper bound is again given by Proposition 41. For the lower bound, we have shown that the 3-colouring problem is NP-hard for triangle-full 4-clique-free graphs. Furthermore, given a triangle-full 4-clique-free graph G , we can construct in polynomial time an automaton \mathcal{A}_G recognising a language L_G (Lemma 44) such that:

- If G is 3-colourable then there is a deterministic generalised Büchi automaton \mathcal{B} with three states recognising L_G (Lemma 34).
- If there is a generalised Büchi automaton \mathcal{B} with three states recognising L_G (in particular, if there is a deterministic one), then G is 3-colourable (Lemma 45).

Hence the 3-colouring problem over triangle-full 4-clique-free graphs reduces to the minimisation problem for deterministic generalised Büchi automata, which is therefore NP-hard. \blacktriangleleft

C.3 NP-completeness for minimising states and colours

We consider the problems of minimising both colours and states simultaneously and show that it is NP-complete for (history-)deterministic generalised (co)Büchi automata. For deterministic generalised Büchi automata, deterministic generalised coBüchi automata, and for history-deterministic generalised Büchi automata, we are able to lift our proof of NP-hardness for state-minimisation to NP-hardness for minimising of states and colours simultaneously, without much difficulty.

► **Theorem 46.** *The following problems are NP-complete:*

1. *Given a deterministic generalised Büchi automaton \mathcal{A} and numbers n and k , is there an equivalent deterministic generalised Büchi automata with at most n states and k colours?*
2. *Given a deterministic generalised coBüchi automaton \mathcal{A} and numbers n and k , is there an equivalent deterministic generalised coBüchi automata with at most n states and k colours?*
3. *Given a history-deterministic generalised Büchi automaton \mathcal{A} and numbers n and k , is there an equivalent history-deterministic generalised Büchi automata with at most n states and k colours?*

Proof. Containment in NP for each of these problems is not quite immediate as if k is encoded in binary, a witness automaton could have exponentially many colours in the size of the input. However, by Lemmas 31 and 32 we know that this cannot happen: if there is an equivalent automaton \mathcal{B} with n states and k colours in the same class of automata, then there is one with polynomially many colours in the number of states and colours of \mathcal{A} . One can thus guess \mathcal{B} and check for simulation between \mathcal{A} and \mathcal{B} both ways to verify language equivalence [42] and whenever applicable, history-determinism [27, Theorem 4.1].

The NP-hardness for minimising states and colours for deterministic and history-deterministic generalised Büchi automata follows from the proofs of Lemmas 34 and 45, from where we can infer that there is a deterministic generalised Büchi automaton \mathcal{B} with k states and k colours recognising L_G if and only if G is k -colourable.

By duality in the deterministic case, this also yields NP-hardness of state and colour minimisation for deterministic generalised coBüchi automata. ◀

For proving the NP-hardness of state and colour minimisation for HD generalised coBüchi automata (Theorem 30), we reduce from the colouring problem for graphs.

In all that follows we will only consider connected graphs, and assume that every vertex has degree at least 2. It is easy to see that the colouring problem remains NP-complete with those assumptions.

We select an arbitrary vertex $v_{\text{init}} \in V$. A *pseudo-path* in G is a (finite or infinite) sequence $v_0 e_0 v_1 e_1 \dots \in (V \cup E)^\infty$ such that $v_i, v_{i+1} \in e_i$ for all i . Note that we allow v_i and v_{i+1} to be equal, hence the term pseudo-path; that is, we allow a pseudo-path to step on an edge without going through it, and come back to the previous vertex. A pseudo-path is *initial* if $v_0 = v_{\text{init}}$. We say that such a pseudo-path *stabilises around* v if it is infinite and there exists i such that for all $j > i$, $v_j = v$, i.e., the pseudo-path eventually stays on the same vertex and just steps on the adjacent edges. We write $\text{Stab}(v)$ for the set of initial pseudo-paths stabilising around v , and we define the language $L_G^{\text{stab}} = \bigcup_{v \in V} \text{Stab}(v)$ over the alphabet $V \cup E$. For $v \in V$, we write $\text{adj}(v)$ for the set of edges $\{e \in E \mid v \in e\}$.

We define the automaton \mathcal{A}_G as follows:

- $Q = V \cup E \cup \{q_{\text{init}}\}$, where q_{init} is a fresh element, which is the initial state,
- $\Sigma = V \cup E$,

- $\Delta = \{(q_{\text{init}}, v_{\text{init}}, v_{\text{init}})\} \cup \{(v, e, e) \in V \times E^2 \mid v \in e\} \cup \{(e, v, v) \in E \times V^2 \mid v \in e\}$,
- the colour of a transition reading $v \in V$ is $V \setminus \{v\}$, the colour of a transition reading $e = \{v_1, v_2\}$ is $V \setminus \{v_1, v_2\}$

This automaton is deterministic and recognises the language $\bigcup_{v \in V} \text{Stab}(v)$. Note that it is not complete: it only reads initial pseudo-paths of G . It can easily be made complete by adding a sink state.

► **Lemma 47.** *The automaton \mathcal{A}_G has an equivalent history-deterministic generalised coBüchi automaton \mathcal{B} with at most $|Q|$ states and k colours if and only if \mathcal{A}_G can be recoloured with k colours.*

Proof. The right-to-left direction is immediate. We now prove the left-to-right direction.

We assumed that every vertex in G has degree at least 2. As there is at most one edge between two vertices. As a consequence, $\text{adj}(v) \neq \text{adj}(v')$ for all $v \neq v' \in V$. This implies that all states of \mathcal{A}_G have distinct residuals.

As \mathcal{B} is history-deterministic, for each residual of the language, it must have a state recognising that residual. Hence \mathcal{B} has exactly $|Q|$ states, one for each residual. It is therefore deterministic, and its states and transitions are isomorphic to \mathcal{A} .

We can thus simply copy the colouring function of \mathcal{B} on the transitions of \mathcal{A} to obtain an equivalent generalised coBüchi automaton. ◀

► **Lemma 48.** *A connected graph G admits a k -colouring if and only if \mathcal{A} can be recoloured with k colours.*

Proof. For the left-to-right direction, let $c: V \rightarrow \{1, \dots, k\}$ be a k -colouring of G . We define the colouring $\text{col}' : \Delta \rightarrow 2^{\{1, \dots, k\}}$ with $\text{col}'(e \xrightarrow{v} v) = \{1, \dots, k\} \setminus \{c(v)\}$ and $\text{col}'(v_1 \xrightarrow{e} v_2) = \{1, \dots, k\} \setminus \{c(v_1), c(v_2)\}$ for all $e = \{v_1, v_2\} \in E$. Let us prove that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Let $w \in \mathcal{L}(\mathcal{A})$, w is an initial pseudo-path and there must exist v such that w stabilises around v . Let $i = c(v)$, ultimately w only visits $v \cup \text{adj}(v)$ and thus it never produces colour i , so $w \in \mathcal{L}(\mathcal{A}')$. Let $w \in \mathcal{L}(\mathcal{A}')$. Again, w is an initial pseudo-path and there must exist i such that w never sees i after some point, meaning that it ultimately only visits $\bigcup_{v \in c^{-1}(i)} \{v\} \cup \text{adj}(v)$. Moreover, as c is a k -colouring of G , $c^{-1}(i)$ is an independent set (that is, no two vertices are connected by an edge), hence w cannot visit infinitely often two distinct vertices from this set without visiting infinitely often an intermediate vertex of a different colour. As a consequence, w must stabilise around some $v \in c^{-1}(i)$, thus $w \in \mathcal{L}(\mathcal{A})$. We have shown that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

For the other direction of the reduction, suppose we have a colouring $\text{col}' : \Delta \rightarrow 2^{\{1, \dots, k\}}$ such that \mathcal{A}' obtained by replacing col with col' in \mathcal{A} is equivalent to \mathcal{A} . Then we define a colouring $c: V \rightarrow \{1, \dots, k\}$ as follows. For all $v \in V$ we choose $c(v) \notin \bigcup_{e \in E, v \in e} \text{col}'(e \xrightarrow{v} v) \cup \text{col}'(v \xrightarrow{e} e)$. It is well-defined as seeing all those transitions infinitely many times yields a run reading a word in $\text{Stab}(v)$, hence this run must be accepting and thus must avoid a colour.

Two neighbours u, v cannot be coloured with the same colour i by c , as otherwise one could read the word $u_v(\text{ev}e')^\omega$ with u_v a pseudo-path ending in v and $e = \{v, v'\}$. This word is not in the language of \mathcal{A}_G , but reading it would yield a run eventually avoiding colour i , hence an accepting run, a contradiction.

We thus obtain that c is a k -colouring of G . ◀

By Lemma 47, G has a k -colouring if and only if \mathcal{A} can be recoloured with k colours without changing its language, if and only if there is an automaton with $\leq |Q|$ states and $\leq k$ colours equivalent to \mathcal{A} .