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

Antonio Casares
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
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Aditya Prakash
University of Warwick, UK

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 → Verification by model checking

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

Digital Object Identifier 10.4230/LIPIcs.CSL.2025.22

Related Version *Long version with appendices*: `https://arxiv.org/abs/2407.18090`

Funding *Antonio Casares*: Supported by the Polish National Science Centre (NCN) grant "Polynomial finite state computation" (2022/46/A/ST6/00072).

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 $(\mathbb{N}, \mathsf{succ})$ [14], 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, 42, 24] and synthesis (given a set of specifications, automatically construct a system satisfying them) [13, 35]. 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 [38].

In 2006, Henzinger and Piterman [26] proposed[1] a model of automata, called *history-deterministic* (HD)[2], presenting a restricted amount of non-determinism so that they exactly 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 [30, 18], the problem of recognising them [4, 8], or extensions to other settings [32, 9, 10].

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 $\mathcal{O}(n \log n)$ [27].

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 [39]. 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 [40]. Multiple other results have backed the idea that transition-based acceptance is a better-suited model; we refer to [16, 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 [15].

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 [21, 25, 41]; for instance, `SPOT`'s LTL-synthesis tool transforms a given LTL formula into a generalised Büchi automaton [22, 33]. Also, many efficient algorithms for their emptiness check have been developed [36, 37, 6].

[1] Similar ideas had been previously investigated by Kupferman, Safra and Vardi [31], and Colcombet studied history-determinism in the context of cost functions [20].

[2] These automata were first introduced under the name *good-for-games* (GFG). Currently, these two notions are no longer used interchangeably, although they coincide in the case of ω-automata. We refer to the survey [11] for further discussions.

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 [41, 29] (which do not yield minimal automata), or by the application of SAT solvers [23, 3]. However, to the best of our knowledge, no theoretical result about the exact complexity of this minimisation problem appears in the literature.

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 [19] 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.

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

Model \ Condition	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)

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 the long version, we also lift the characterisation based on the G_2 game from (co)Büchi automata to generalised ones (a similar remark was suggested in the conclusion of [8]).

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, 39]), 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 no two transitions between two states over the

same input letter (see [18, 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. A *factor* of a word w is a word u such that there exist words x, y with $w = xuy$. For an infinite word $w \in \Sigma^\omega$, we denote $\mathsf{Inf}(w)$ the set of letters occurring infinitely often in w.

2.1 Automata

We let Σ be a finite alphabet. An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_{\text{init}}, \Delta, \Gamma, \mathsf{col}, W)$, 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*, $\mathsf{col}\colon \Delta \to \Gamma$ 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 $\mathsf{col}((p, a, q)) = c$.

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

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

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

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

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

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 $\mathsf{col}((p, a, q)) = \{1\}$. 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 [34].

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 $\sigma : \Sigma^+ \to \Delta$ such that for all words $w = a_0 a_1 \cdots \in \Sigma^\omega$, the sequence $\sigma^*(w) = \sigma(a_0)\sigma(a_0a_1)\sigma(a_0a_1a_2)\cdots \in \Delta^\omega$ (called the *run induced by σ over w*) satisfies:

1. $\sigma^*(w)$ is a run on w in $\mathcal{A}$,
2. if $w \in \mathcal{L}(\mathcal{A})$, then $\sigma^*(w)$ 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) [31].

▶ **Example 2** (From [17, Ex. 2.3])**.** Let $\Sigma = \{a, b, c\}$ and $L = \{w \in \Sigma^\omega \mid \{b, c\} \not\subseteq \mathsf{Inf}(w)\}$, 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 prunning*, 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 at 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 at 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.

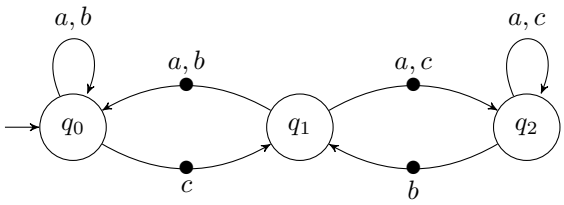

Figure 1 A history-deterministic coBüchi automaton recognising the language $L = \{w \in \Sigma^\omega \mid \{b, c\} \not\subseteq \mathsf{Inf}(w)\}$. CoBüchi transitions are represented with a dot on them.

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 $\mathsf{Res}(L)$ 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 $R \in \mathsf{Res}(L)$ 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 $\phi\colon Q_1 \to Q_2$ 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 $\phi_\Delta\colon \Delta_1 \to \Delta_2$ 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 $\phi\colon \mathcal{S}_1 \to \mathcal{S}_2$.

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 Lemma 15), 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.

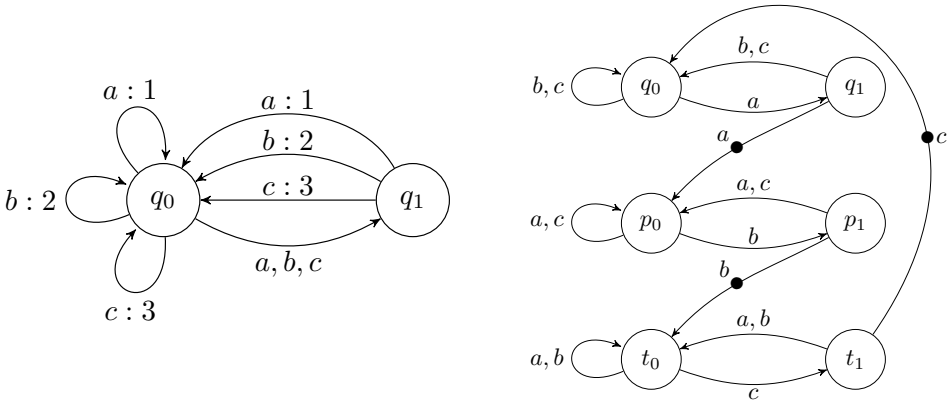

Figure 2 On the left, a history-deterministic generalised coBüchi automaton recognising the language L_3 of words eventually avoiding factor xx for some letter x. On the right, a minimal deterministic coBüchi automaton for the same language (coBüchi transitions have a dot on them). In both cases, the 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 $\mathsf{genB}_{[k]}$ with $\mathsf{genCoB}_{[k]}$ 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 the long version we prove that any non-deterministic generalised Büchi automaton recognising $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$ has at least 3 states). Relatedly, for the non-generalised conditions, 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 [30], using the link between complementation and determinisation of HD automata from [7, Thm 4].

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

The idea of this construction is to define a particular deterministic coBüchi automaton recognizing genCoB_C, and to associate it to the input generalised coBüchi automaton via a cascade composition (defined below) in order to obtain the wanted coBüchi automaton. We will now detail these different steps.

Deterministic coBüchi automaton for genCoB_C. Let $C = \{1, 2, \cdots, k\}$ be a set of k colours; for convenience, in the context of colours, we will use the symbol $+$ to denote addition modulo k, in particular, $k + 1 = 1$. We build a deterministic coBüchi automaton $\mathcal{D}_C^{\mathrm{coB}}$ 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 $i \in X$, for all $X \in \Gamma$. The initial state is arbitrary.

We claim that the automaton $\mathcal{D}_C^{\mathrm{coB}}$ recognises the language genCoB_C. First, we remark that the accepting runs of $\mathcal{D}_C^{\mathrm{coB}}$ are exactly those that eventually remain forever in a state q_i. Let $w = w_1 w_2 \cdots \in 2^C$. If w is accepted by $\mathcal{D}_C^{\mathrm{coB}}$, then the run on w eventually stays in a q_i, so w eventually does not contain colour i, and $w \in \mathsf{genCoB}_C$. Conversely, if w is rejected by $\mathcal{D}_C^{\mathrm{coB}}$, 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 $\mathcal{D}_C^{\mathrm{B}}$ recognising the language genB_C, simply by changing the acceptance condition of genCoB_C to genB_C.

▶ Remark 7. The automaton $\mathcal{D}_C^{\mathrm{coB}}$ has k states, but exponentially many transitions. This is made possible by the fact that its alphabet Γ is exponential in the number of colours k.

Cascade composition of automata. Let $\mathcal{A} = (Q_\mathcal{A}, \Sigma_\mathcal{A}, q_{\mathrm{init}}^\mathcal{A}, \Delta_\mathcal{A}, \Gamma_\mathcal{A}, \mathsf{col}_\mathcal{A}, W_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, \Sigma_\mathcal{B}, q_{\mathrm{init}}^\mathcal{B}, \Delta_\mathcal{B}, \Gamma_\mathcal{B}, \mathsf{col}_B, W_B)$ 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 *cascade composition* of $\mathcal{A}$ and $\mathcal{B}$ is the automaton over $\Sigma_\mathcal{A}$ defined as:

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

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 $\mathsf{col}_\mathcal{A}$ 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 $\mathcal{D}_C^{\mathrm{coB}}$. 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 $\mathcal{D}_C^{\mathrm{coB}} \circ \mathcal{A}$ is a coBüchi automaton equivalent to $\mathcal{A}$ which is (history-)deterministic if and only if $\mathcal{A}$ is. Moreover, $\mathcal{D}_C^{\mathrm{coB}} \circ \mathcal{A}$ 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 $\mathcal{D}_C^{\mathrm{coB}} \circ \mathcal{A}$ 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 $\mathcal{D}_C^{\mathrm{coB}} \circ \mathcal{A}$ from $\mathcal{A}$ in polynomial time in the size of its representation. Indeed, we just need to compose $\mathcal{A}$ with the restriction of $\mathcal{D}_C^{\mathrm{coB}}$ 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 [30, 4, 8]. 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 the long version.

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 $\mathcal{A}_{\mathsf{safe}}$ 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 $\mathcal{A}_{\mathsf{safe}}$; 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 $\mathsf{Safe}(\mathcal{A})$ be the set of safe components of $\mathcal{A}$.

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

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

▶ **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 $\mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^{q_0}) = \{w \in \Sigma^\omega \mid w \text{ does not contain the factor } aa\}$.

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 $\mathcal{A}_{\mathsf{safe}}$ 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 $\mathcal{A}$ can be assumed to be semantically deterministic. This means that different choices made by a resolver from the same state with different histories must be consistent with the residual, i.e. must lead to states accepting the same language, which is the language $u^{-1}L(\mathcal{A})$ after reading a word u. Kuperberg and Skrzypczak showed the more involved result that we can moreover assume safe determinism [30]. All in all, we have:

▶ **Lemma 13** ([30]). *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 [30], 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 ($\mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^q) \subseteq \mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^p)$ 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 $\mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^q) = \mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^p)$ implies $q = p$.

▶ **Example 14.** 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 $\mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^{q_1}) = \emptyset \subseteq \mathcal{L}_{\mathsf{Safe}}(\mathcal{A}^{q_2})$, but q_1 and q_2 appear in different safe components.

▶ **Lemma 15** ([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 $\eta\colon \mathsf{Safe}(\mathcal{A}_{\min}) \to \mathsf{Safe}(\mathcal{A})$ such that for every safe component $\mathcal{S} \in \mathsf{Safe}(\mathcal{A}_{\min})$, it holds that $|\mathcal{S}| \leq |\eta(\mathcal{S})|$.*

Minimality of such automata directly follows:

▶ **Corollary 16.** *Let $\mathcal{A}_{\min}$ be a nice, safe minimal and safe centralised HD coBüchi automaton. Then, the number of states of $\mathcal{A}_{\min}$ is minimal among all HD coBüchi automata recognising the same language.*

▶ **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 $\mathcal{A}_L^{\mathsf{coB}}$. Moreover, such an automaton $\mathcal{A}_L^{\mathsf{coB}}$ 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 17, we obtain that we can build in polynomial time the minimal HD coBüchi automaton $\mathcal{A}_L^{\mathsf{coB}}$ for L. Let $\mathsf{Safe}(\mathcal{A}_L^{\mathsf{coB}}) = \{\mathcal{S}_1, \dots, \mathcal{S}_k\}$ be an enumeration of the safe components of $\mathcal{A}_L^{\mathsf{coB}}$, 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 $\mathcal{A}_L^{\mathsf{genCoB}}$ of size $n_{\max} = \max_{1 \le i \le k} |S_i|$ and using k output colours.

Intuitively, $\mathcal{A}_L^{\mathsf{genCoB}}$ will be the full automaton (it contains transitions between all pairs of states, for all input letters). Since $|S_i| \le n_{\max}$, 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 $\mathcal{A}_L^{\mathsf{coB}}$. That is, the transitions of $\mathcal{A}_L^{\mathsf{genCoB}}$ that are "safe-for-colour i" will be exactly those in $\phi_i(\mathcal{S}_i)$.

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

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

Finally, we define the colour labelling $\mathsf{col} \colon \Delta \to \Gamma$. For each $1 \le i \le k$, let $\phi_i \colon \mathcal{S}_i \to (Q, \Delta)$ be any injective morphism of automaton structures (such a morphism exists since $|\mathcal{S}_i| \le n_{\max}$ 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, $\mathsf{col}(e) = \{i \mid \phi_i^{-1}(e) = \emptyset\}$.

▶ Remark 18. We remark that this colour labelling uses some arbitrary choices, namely, the way we map the different safe components of $\mathcal{A}_L^{\mathsf{coB}}$ 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 $\mathcal{A}_L^{\mathsf{genCoB}}$ 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 $\mathcal{A}_L^{\mathsf{coB}}$ (on the right) and superposing them in a 2-state automaton. To simplify its presentation, we have removed some unnecessary transitions of $\mathcal{A}_L^{\mathsf{genCoB}}$, 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 $\mathcal{A}_L^{\mathsf{genCoB}}$ is history-deterministic and recognises L.*

Proof sketch. If w admits an accepting run ρ in $\mathcal{A}_L^{\mathsf{genCoB}}$, 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 $\mathcal{A}_L^{\mathsf{coB}}$, so $w \in L$. This is where we use prefix-independence: as soon as there is a witness that a suffix of w is also a suffix of a word in L, then the whole word w is in L as well.

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

Proof. Let $C = \{1, \dots, k\}$ be the set of output colours used by the acceptance condition of $\mathcal{B}$ and let $\mathcal{D}_C^{\mathrm{coB}}$ be the coBüchi automaton recognising genCoB_C presented in Section 3.3. By Corollary 9, $\mathcal{D}_C^{\mathrm{coB}} \circ \mathcal{B}$ is a history-deterministic automaton recognising L. Moreover, the states of $\mathcal{D}_C^{\mathrm{coB}} \circ \mathcal{B}$ are a disjoint union $Q_1 \uplus Q_2 \uplus \cdots \uplus Q_k$ 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 $\mathcal{D}_C^{\mathsf{coB}} \circ \mathcal{B}$ is included in some Q_i, so $|\mathcal{S}| \leq |\mathcal{B}|$. By Lemma 13, we can turn $\mathcal{D}_C^{\mathsf{coB}} \circ \mathcal{B}$ into a nice HD coBüchi automaton $\mathcal{B}'$ satisfying that none of its safe components is larger than $|\mathcal{B}|$.

By Lemma 15 there is an injection $\eta\colon \mathsf{Safe}(\mathcal{A}_L^{\mathsf{coB}}) \to \mathsf{Safe}(\mathcal{B}')$ such that $|\mathcal{S}| \leq |\eta(\mathcal{S})|$ for all safe component $\mathcal{S}$ of $\mathcal{A}_L^{\mathsf{coB}}$. In particular, if $\mathcal{S}_{\max}$ is a safe component of maximal size in $\mathcal{A}_L^{\mathsf{coB}}$, we obtain: $|\mathcal{A}_L^{\mathsf{genCoB}}| = |\mathcal{S}_{\max}| \leq |\eta(\mathcal{S}_{\max})| \leq |\mathcal{B}|$. ◀

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 the prefix-independence assumption). 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 $\mathcal{A}_L^{\mathsf{coB}}$ in polynomial time. We show how to convert it to an equivalent HD generalised coBüchi automaton $\mathcal{A}_L^{\mathsf{genCoB}}$ of minimal size.

Let $R_1, R_2, \cdots, R_m$ be all the distinct residual languages of L, i.e., languages of the form $u^{-1}L$ for some finite word $u \in \Sigma^*$. The case $m = 1$ corresponds to the prefix-independent case treated in Section 4.2. We note that these residuals induce a partition of the states of $\mathcal{A}_L^{\mathsf{coB}}$ into $Q^{R_1}, \ldots, Q^{R_m}$, 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 $\mathcal{A}_L^{\mathsf{coB}}$. Let $\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k$ be the safe components of $\mathcal{A}_L^{\mathsf{coB}}$, 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 $\mathcal{A}_L^{\mathsf{coB}}$. 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 $\mathcal{A}_L^{\mathsf{genCoB}}$ with $n_1 + n_2 + \cdots + n_m$ states. Towards this, for each residual language R_j, let $P_j = \{p_j^1, p_j^2, \cdots, p_j^{n_j}\}$ be a set of n_j elements. The automaton $\mathcal{A}_L^{\mathsf{genCoB}} = (Q, \Sigma, q_{\text{init}}, \Delta, \Gamma, \mathsf{col}, \mathsf{genB})$ is given by:

- $Q = P_1 \uplus P_2 \uplus \cdots \uplus P_m$.
- $q_{\text{init}} = p_1^1$ (any state corresponding to the residual of the initial state of $\mathcal{A}_L^{\mathsf{coB}}$ would work).
- Let (q, a, q') be a transition in $\mathcal{A}_L^{\mathsf{coB}}$, 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 $p' \in P_{j'}$.
- $\Gamma = 2^{\{1,\ldots,k\}}$.

One way of picturing $\mathcal{A}_L^{\mathsf{genCoB}}$ 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 $\mathsf{col}\colon \Delta \to \Gamma$. Informally, each safe component $\mathcal{S}_i$ is mapped into $\mathcal{A}_L^{\mathsf{genCoB}}$ 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 $\mathcal{A}_L^{\mathsf{coB}}$ corresponds to a run in $\mathcal{A}_L^{\mathsf{genCoB}}$ that eventually avoids colour i.

More formally, for $i \in [1, k]$, let $\phi_i\colon \mathcal{S}_i \to \mathcal{A}_L^{\mathsf{genCoB}}$ 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 $\mathcal{A}_L^{\mathsf{genCoB}}$ 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 $\mathsf{col}(e)$ contains exactly the colours in $\{i \mid \phi_i^{-1}(e) = \emptyset\}$.

▶ Remark 22. We remark that the automaton $\mathcal{A}_L^{\mathsf{genCoB}}$ obtained in this way uses a polynomial number of output colours in the size of the minimal HD coBüchi automaton $\mathcal{A}_L^{\mathsf{coB}}$ (see also long version). More precisely, the number k of colours is the number of safe components of $\mathcal{A}_L^{\mathsf{coB}}$. However, this number of colours 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 $\mathcal{A}_L^{\mathsf{genCoB}}$ is history-deterministic and recognises L.*

In particular, the resolver for $\mathcal{A}_L^{\mathsf{genCoB}}$ is defined as in Proposition 20: it follows the different safe components of $\mathcal{A}_L^{\mathsf{coB}}$ in a round-robin fashion, by trying to avoid colour some colour i while moving in $\phi_i(S_i)$, then if colour i is seen it switches to $i' = (i+1) \mod k$, etc.

We explain how to obtain the minimality of $\mathcal{A}_L^{\mathsf{genCoB}}$, stated below. We reduce to the prefix-independent case, using a technique from [12].[3] Full proofs can be found in the long version.

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

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

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

That is, if $\mathcal{A}$ is a semantically deterministic automaton with states Q, then $\Sigma\restriction_R$ is the set of words that connect states in Q^R. Note that in general $\Sigma\restriction_R$ may be infinite, however this is harmless in this context, and we will freely allow ourselves to talk about automata over infinite alphabets. Also, $\Sigma\restriction_R$ 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 the long version.

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

▶ Remark 25. For every residual R, $L\restriction_R$ is a prefix-independent language. It corresponds to infinite words whose letters are in $\Sigma\restriction_R$ that is, going from Q^R to Q^R, and eventually avoid seeing some colour on such paths. The prefix-independence of $L\restriction_R$ follows from the fact that $L\restriction_R$ has only itself as residual, on alphabet $\Sigma\restriction_R$.

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 $\mathcal{A}\restriction_R$ to be the generalised coBüchi automaton over $\Sigma\restriction_R$ 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 $w \in \Sigma\restriction_R$, $X \in 2^{\{1,\dots,k\}}$, if there is a path from q to p labelled w and producing the set of colours X in $\mathcal{A}$.

[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 Lemma 15, stating that the injection η satisfies that, for every residual R and safe component $\mathcal{S}$ of $\mathcal{A}_L^{\mathsf{coB}}$, $|\mathcal{S} \cap R| \leq |\eta(\mathcal{S}) \cap R|$. The proof of Abu Radi and Kupferman [2] does indeed lead to this result, but it is not explicitly stated in this form.

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

Using the fact that (safe) paths between states in Q^R are the same in $\mathcal{A}$ or in $\mathcal{A}\restriction_R$, combined with Lemma 15, we can prove:

▶ **Lemma 27.** *$\mathcal{A}_L^{\mathsf{coB}}\restriction_R$ is a minimal HD coBüchi automaton recognising $L\restriction_R$. 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: $|\mathcal{B}| \geq n_1 + \cdots + n_m = |\mathcal{A}_L^{\mathsf{genCoB}}|$.

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.

▶ **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 deterministic 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 [19], itself inspired from [28]. Details can be found in the long version.

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 the long version.

▶ **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 [40, 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 $\mathcal{A}_G$, and
- if G is not 3-colourable then there is no automaton $\mathcal{B}$ (deterministic or not) with 3 states equivalent to $\mathcal{A}_G$.

This establishes the hardness of state-minimisation for deterministic and history-deterministic automata simultaneously. The full proof is in the long version. 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 $c : V \to \{1, \dots, k\}$ 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. ◀

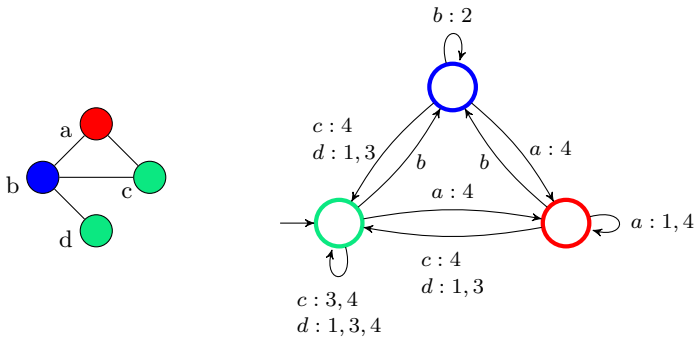

Figure 3 A graph with a 3-colouring, and the corresponding automaton as defined in Lemma 34. The output colours a, b, c, d have been replaced by $1, 2, 3, 4$ for readability.

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.

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 Baarir 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. Seminator 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, Orna Kupferman, and Michal Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100. Springer, 2013. doi:10.1007/978-3-642-39212-2_11.

8 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.

9 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.

10 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.

11 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.

12 Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove. 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.

13 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.

14 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.

15 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.

16 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.

17 Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. From Muller to Parity and Rabin Automata: Optimal Transformations Preserving (History) Determinism. *TheoretiCS*, Volume 3, April 2024. doi:10.46298/theoretics.24.12.

18 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.

19 Antonio Casares and Corto Mascle. The complexity of simplifying ω-automata through the alternating cycle decomposition. *CoRR*, abs/2401.03811, 2024. arXiv:2401.03811, doi:10.48550/arXiv.2401.03811.

20 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.

21 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.

22 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.

23 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.

24 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.

25 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.

26 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer Science Logic*, pages 395–410, 2006. doi:10.1007/11874683_26.

27 John E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford University, 1971. doi:10.5555/891883.

28 Christopher Hugenroth. Zielonka DAG acceptance, regular languages over infinite words. In *DLT*, 2023.

29 Sudeep Juvekar and Nir Piterman. Minimizing generalized Büchi automata. In *CAV*, pages 45–58, 2006. doi:10.1007/11817963_7.

30 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.

31 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.

32 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. In *LICS*, pages 689–702, 2020. doi:10.1145/3373718.3394737.

33 Thibaud Michaud and Maximilien Colange. Reactive synthesis from LTL specification with Spot. In *SYNT@CAV*, Electronic Proceedings in Theoretical Computer Science, 2018.

34 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.

35 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989. doi:10.1145/75277.75293.

36 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.

37 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.

38 Schmuel Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327, 1988. doi:10.1109/SFCS.1988.21948.

39 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.

40 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.

41 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.

42 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994. doi:10.1006/inco.1994.1092.