History-Deterministic Buchi Automata are Succinct

Antonio Casares
University Kaiserslautern-Landau, Germany

Aditya Prakash
Aix Marseille Université, CNRS, LIS, Marseille, France

K. S. Thejaswini
Université Libre de Bruxelles, Belgium

Abstract

We describe a history-deterministic Büchi automaton that has strictly less states than every language-equivalent deterministic Büchi automaton. This solves a problem that had been open since the introduction of history-determinism and actively investigated for over a decade.

Our example automaton has 65 states, and proving its succinctness requires the combination of theoretical insights together with the aid of computers.

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

Keywords and phrases History-deterministic automata, Succinctness, Büchi automata

Funding *Antonio Casares*: Partially supported by Deutsche Forschungsgemeinschaft (grant number 522843867) and European Research Council (grant number 101089343).
Aditya Prakash: Supported by ANR grant QuaSy (ANR 23-CE48-0008)
K. S. Thejaswini: A part of project was done when the author was employed by Institute of Science and Technology Austria and received funding from the European Research Council (ERC), grant agreement No 101020093.

1 Introduction

Automata over infinite words are a well-established tool with applications in the verification and synthesis of non-terminating systems [Kup18, EKV21]. A major bottleneck in these applications is the exponential cost of determinising automata over infinite words. To circumvent this, recent research has focused on history-deterministic automata (HD automata hereafter): a class of "mildly" nondeterministic automata that offer the algorithmic benefits of determinism without the full cost of determinisation. Formally, an automaton is history-deterministic if there is a strategy resolving the nondeterministic choices on-the-fly without guessing the future. HD automata were introduced by Henzinger and Piterman [HP06] under the name of good-for-games automata, as they are exactly the class of automata that can be composed with infinite duration games on graphs while preserving the winner. This is the key property that enables their use in verification and synthesis (see, e.g., [BL23, page 22]). Equivalent definitions of history determinism appear in the work of Kupferman, Safra, and Vardi [KSV96] and were unified by Boker, Kuperberg, Kupferman, and Skrzypczak [BKKS13]. Moreover, Colcombet independently defined the concept of history-determinism in the setting of cost automata [Col09].

In this work, we focus on Büchi and coBüchi automata. A Büchi automaton is an automaton with some of its transitions marked as *significant*. A word is accepted by a Büchi automaton if it admits a run that contains infinitely many significant transitions. CoBüchi automata are the dual of Büchi automata: a word is accepted if it admits a run which contains only finitely many significant transitions.

We exemplify the above notions in the next example.

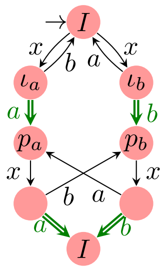


Figure 1 A HD Büchi automaton $\mathcal{A}_{\text{BKKS}}$ that is not determinisable by pruning. States labelled I are identified as the same state. Double green arrows represent significant transitions.

▶ **Example 1.** The automaton $\mathcal{A}_{\text{BKKS}}$ described in Figure 1—a version of [BKKS13, Fig.2]—accepts the language of words of the form $(xa + xb)^\omega$ that contain at least one of the strings $xaxa$ or $xbxb$ infinitely often. This automaton is HD. The only nondeterministic choice is on I over the letter x. A strategy for resolving this nondeterminism is simple: we alternate between choosing the transitions $I \xrightarrow{x} \iota_a$ and $I \xrightarrow{x} \iota_b$. If the input word is in the language, it must eventually contain $xaxa$ or $xbxb$, allowing us to progress to states p_a or p_b and visit an accepting transition (infinitely often).

As mentioned before, HD automata constitute an alternative for the use of deterministic automata in various applications. In procedures such as LTL-synthesis, the main bottleneck is the size of the resulting deterministic automata [ERS24]. One motivation for the use of HD automata is the potential reduction in size. For this reason, a central theme in the study of HD automata is *succinctness*: can HD automata be smaller than their deterministic counterparts?

For coBüchi automata, the landscape is well-understood. We know that HD coBüchi automata can be exponentially smaller than the smallest language-equivalent deterministic ones [KS15]. Moreover, Abu Radi and Kupferman showed that HD coBüchi automata (with transition-based acceptance) can be minimised in polynomial time, leading to a canonical form for languages recognised by coBüchi automata [AK22]. These theoretical results have subsequently enabled canonical representations of ω-regular languages [ES22, Ehl25, CLW26], algorithms for learning HD coBüchi automata [LW25], and practical implementation efforts [EK24a, EK24b].

In contrast, the situation for Büchi automata has remained elusive. While it is known that every HD Büchi automaton admits an language-equivalent deterministic Büchi automaton of quadratic size [KS15], it is unknown whether this bound could be matched or improved. Actually, it was not even known if every HD Büchi automaton admit an equivalent deterministic Büchi automaton of the same size. In this work, we tackle the following question.

HD-Büchi succinctness problem. *Is there an HD Büchi automaton that is strictly smaller than every language-equivalent deterministic Büchi automaton?*

In fact, for several years since Henzinger and Piterman's introduction of HD automata in 2006 [HP06], it was not known whether HD Büchi automata could simply be determinised by pruning transitions (as is the case for HD automata over finite words [Col12, Proposition 14]). This was settled in the negative by Boker et al. [BKKS13] by the example 7-state automaton in Figure 1. However, there is an equivalent deterministic automaton with only 4 states (see Figure 3a). The computational complexity of the determinisation of HD Büchi automata was only recently proved to be in polynomial time by two independent works [AJP24, AKL24].

In recent years, the research community working on HD automata has been very active, and recent results—for example the works by Abu Radi and Kupferman [AK22] or by Lehtinen and Prakash [LP25]—have gained a lot of insight on the structure of HD automata. The fact that this question remained open points to a gap of our understanding of the structure of HD and deterministic Büchi automata.

Several works raise the above question on the succinctness of HD Büchi automata. A conjecture stating that all HD parity automata (and in particular HD Büchi automata) are determinisable by pruning was stated by Colcombet in 2012 [Col12, Conj. 8]. The succinctness question was asked in 2013 [BKKS13] (for HD automata with various acceptance conditions) and solved in 2015 for coBüchi automata [KS15]. The succinctness question for Büchi automata is discussed in various recent works [AK22, p.30], [AJP24, pages 3 and 16], and [AKL24, p.69]. For the subclass of Muller languages (languages that can be described by a Boolean condition on the letters that appear infinitely often), Casares et al. showed that HD parity and Büchi automata are not more succinct than deterministic ones [CCFL24, Cor. 4.16], while Casares, Colcombet, and Lehtinen showed that HD Rabin automata can be exponentially smaller than deterministic ones [CCL22, Thm. 21]. The PhD thesis of Prakash also stated a conjecture implying that every HD Büchi automaton admits an equivalent deterministic Büchi automaton of the same size [Pra25, Conj. 2] (see Conjecture 5).

In this paper, we answer the succinctness question positively.

▶ **Theorem 2.** *There is a history-deterministic Büchi automaton that has strictly fewer states than every language-equivalent deterministic Büchi automaton.*

Concretely, we provide an HD Büchi automaton with 65 states and prove that every language-equivalent deterministic automaton requires at least 66 states.

Verifying this gap is non-trivial. Although tools like SPOT exist for minimising deterministic transition-based Büchi automata [DRC$^+$22], they use SAT solvers. Our attempt to feed the full 66-state deterministic automaton to such a solver was perhaps overly optimistic; the resulting CNF formula grew large enough to exhaust memory long before it exhausted the search space and, to the best of our knowledge, the solver is still pondering over the existence of a 65-state deterministic automaton. Furthermore, the minimisation of deterministic Büchi automata is an NP-complete problem for both state-based acceptance [Sch10] or transition-based acceptance [AE25].

Hence, we construct our witness automaton in such a way that most of the states of the automaton are essential and cannot be minimised. This is done by leveraging structural insights regarding HD coBüchi automata, drawing primarily on the work of Abu Radi and Kupferman [AK22]. We use their characterisation of minimality for HD coBüchi automata to reason about the minimality of a deterministic coBüchi automaton recognising the complement language.

Even with these theoretical reductions, we are left to prove that the smallest DFA separating two specific languages on finite words requires at least 5 states. To establish this lower bound, we employ the learner tool DFAMiner [DLS24]. This tool formulates the existence of a 4-state separator as a SAT instance and then proves its unsatisfiability, thereby making our proof computer-aided.

Organisation. After introducing the necessary preliminaries in Section 2, we give some intuition and first steps towards the construction of the automaton in Section 3. This intuition is given by considering some conjectures and constructing automata to disprove them. This section serves the following two purposes.

- First, it allows to give a more gentle introduction to the general example and the techniques to prove that a Büchi automaton is history-deterministic.
- Secondly, it provides some insights on what are some necessary properties that a succinct HD Büchi automata must satisfy. This explains why small succinct automata are challenging to find.

We describe our full 65-state example in Section 4, by combining the ideas from Section 3. Finally, in Section 5 we present the proof of succinctness for Theorem 2; in this section we also include encodings of the automata over finite words that we use for DFAminer.

2 Preliminaries

For a natural number i, we use $[i]$ to denote the set $\{1, 2, \ldots, i\}$.

2.1 Automata and history-determinism

An *alphabet*, often denoted by Σ, is a finite set.

Automata. A nondeterministic Büchi (resp. coBüchi) automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ consists of a finite set of *states* Q, an *initial state* $q_0 \in Q$, a set of *transitions* $\Delta \subseteq Q \times \Sigma \times Q$ and a subset $F \subseteq \Delta$ of *significant* transitions. We denote transitions (p, a, q) by $p \xrightarrow{a} q$, and transitions (p, a, q) that are significant by $p \stackrel{a}{\Rightarrow} q$.

A *run* of $\mathcal{A}$ over an infinite word w in Σ^ω is an infinite path ρ of the automaton starting at q_0 such that the edges of ρ are labelled by the letters of w in sequence. If $\mathcal{A}$ is a Büchi automaton, then ρ is an *accepting run* if ρ contains infinitely many significant transitions. Dually, if $\mathcal{A}$ is a coBüchi automaton, then ρ is an *accepting run* if ρ contains finitely many significant transitions. Infinite runs that are not accepting are dubbed *rejecting*.

In the rest of the paper, the word automaton means either a nondeterministic Büchi automaton or a nondeterministic coBüchi automaton, unless mentioned otherwise. A *subautomaton* of $\mathcal{A}$ is an automaton obtained by removing some transitions of $\mathcal{A}$.

A word w is *accepted* by an automaton $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w, and *rejected* otherwise. The *language* of $\mathcal{A}$ is the set of words that are accepted by $\mathcal{A}$, which we denote as $L(\mathcal{A})$. Two automata $\mathcal{A}$ and $\mathcal{B}$ are *language-equivalent* if they have the same language.

For a state q in $\mathcal{A}$, we write $(\mathcal{A}, q)$ to denote the automaton where we set q to be the initial state. We say that two states p, q of $\mathcal{A}$ are *language-equivalent* if $L(\mathcal{A}, p) = L(\mathcal{A}, q)$.

We say that an automaton is *deterministic* if, from every state and for every letter, there is at most one outgoing transition from that state on that letter.

History-determinism. A *resolver* for an automaton $\mathcal{A}$ is a function of type $\Delta^* \times \Sigma \to \Delta$ that takes as input a finite run in $\mathcal{A}$ and a letter a, and outputs a transition on a from the endpoint of the run. This way, a *resolver induces* a run on every word in Σ^*. We say that a resolver for $\mathcal{A}$ is an *HD-resolver* if it induces an accepting run on every word in the language of $\mathcal{A}$. We say that an automaton is *history-deterministic*, HD for short, if there is an HD-resolver for it.

We refer to Example 1 for an example of an HD automaton. We next give a non-example.

▶ **Example 3.** In Figure 2 we show a Büchi automaton that is not history-deterministic. This automaton accepts the words that eventually only contain the letter a. However, to resolve the nondeterminism on letter a, the automaton needs to guess the future. For every strategy deciding to go to the state on the right after a finite prefix, there is an extension of the form ba^ω so that this strategy would reject a word that is in the language.

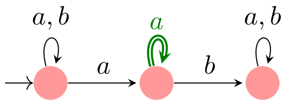


Figure 2 A Büchi automaton that is not HD.

Interpretation of figures. Throughout the paper, we include many figures for automata, some of them representing automata with more states than we would like. To represent these automata, we use some shorthands. First, a state may appear several times in a figure; the final automaton is the result of merging the states with the same name. When many states have the same source (resp. target) over the same letter σ, we may surround these states by a box, and use transitions outgoing from (resp. incoming to) the box. For example, a transition on σ from a box to a state q means that from all states of the box, the σ-transition goes to q. As in the text, double green arrows represent significant transitions for Büchi automata. (For coBüchi automata, significant transitions will be double arrows, but red. These rarely occur.) Sometimes we use different colours and shapes or sizes for states; this is mostly for aesthetic purposes (in Figures 5, 6, and 7, colour differentiation also serves to identify language-equivalent states).

2.2 Structure of HD Büchi automata

Kuperberg and Skrzypcza's quadratic determinisation construction of HD Büchi automata [KS15] first involved proving that every HD Büchi automaton can be simplified so that it satisfies certain properties. Their analysis was then refined later by Acharya, Jurdziński, and Prakash [AJP24]; we will use the presentation from Prakash's PhD thesis [Pra25] in this work. We begin by defining notions that are useful towards this.

Simulation relation. Simulation is a relation between two automata that is coarser than language inclusion, that is, if $\mathcal{B}$ simulates $\mathcal{A}$, then $L(\mathcal{A}) \subseteq L(\mathcal{B})$ [HKR02]. Concretely, the simulation relation can be described as a game between two players Adam and Eve, called the simulation game. The *simulation game of automaton $\mathcal{A}$ by automaton $\mathcal{B}$* starts with an Eve's token at the initial state of the automaton $\mathcal{B}$ and Adam's token at the initial state of $\mathcal{A}$. In each round,

1. Adam selects a letter and moves his token along a transition on that letter in $\mathcal{A}$,
2. Eve moves her token along a transition on that letter in automaton $\mathcal{B}$.

Consequently, in the limit of such a play, Adam constructs a run on his token in $\mathcal{A}$, and Eve constructs a run on her token in $\mathcal{B}$, both on the same word chosen by Adam. Eve *wins* if either Eve's run is accepting or Adam's run is rejecting. If Eve has a winning strategy in this game, then we say that automaton $\mathcal{B}$ *simulates* automaton $\mathcal{A}$. For two states p and q in an automaton $\mathcal{A}$, we will often write q simulates p in $\mathcal{A}$ to say that $(\mathcal{A}, q)$ simulates $(\mathcal{A}, p)$.

Reach-covering. A *reachability automaton* is a Büchi automaton where all of its significant transitions lead to an *accepting sink state* that has a significant self-loop on every letter.

For every Büchi automaton $\mathcal{A}$, we use $\texttt{reach}(\mathcal{A})$ to denote the reachability automaton that is obtained by redirecting all the significant transitions of $\mathcal{A}$ to an accepting sink-state, and by not modifying all the transitions that are not significant.

We say that q is a *reach-deterministic state* in $\mathcal{A}$ if the accessible portion of $(\texttt{reach}(\mathcal{A}), q)$ is *deterministic*, i.e., every word w has an unique run from q in $\texttt{reach}(\mathcal{A})$.

We say that $\mathcal{A}$ has *reach-covering* if for every state p in $\mathcal{A}$, there is a reach-deterministic state q of $\mathcal{A}$ such that the following two conditions hold.

1. p and q are language-equivalent in $\mathcal{A}$.
2. p simulates q in $\texttt{reach}(\mathcal{A})$.

Note that, since reach-deterministic states simulate themselves in $\texttt{reach}(\mathcal{A})$, $\mathcal{A}$ has reach-covering if and only if the above conditions hold for every state p that is not reach-deterministic.

Semantic determinism. We say that an automaton $\mathcal{A}$ is *semantically deterministic*, SD for short, if outgoing transitions on the same letter from language-equivalent states lead to language-equivalent states. That is, if p, q are language-equivalent states in $\mathcal{A}$ and $p \xrightarrow{a} p', q \xrightarrow{a} q'$ are transitions on some letter a, then p' and q' are also language-equivalent.

Normal form. Given a Büchi automaton $\mathcal{A}$, consider the graph consisting only of non-significant transitions of $\mathcal{A}$. The automaton $\mathcal{A}$ is *normal* if every non-significant transition is in some SCC of this graph. We note that every Büchi automaton can be converted into a language-equivalent normal automaton by making significant all non-significant transitions of $\mathcal{A}$ changing of SCC; this transformation does not introduce any accepting cycles, and thus, does not increase the language of the automaton.

Simplified Büchi automata. We say that a Büchi automaton $\mathcal{A}$ is *simplified* if $\mathcal{A}$ is semantically deterministic, normal, and has *reach-covering.* Acharya, Jurdziński, and Prakash showed that every HD Büchi automaton can be *simplified* in polynomial time without increasing the number of states [AJP24] (see also [Pra25, Lemma 5.8]). We will use their result that every simplified Büchi automaton is history-deterministic as well, in order to prove history-determinism of various Büchi automata.

▶ **Lemma 4** ([Pra25, Lemma 5.7]). *Every simplified Büchi automaton is history deterministic.*

Proof sketch. The HD-resolver for a simplified Büchi automaton $\mathcal{A}$, when at state p, simulates an auxiliary run from a reach-deterministic state q such that p and q are language-equivalent in $\mathcal{A}$ and p simulates q in $\texttt{reach}(\mathcal{A})$. Note that the auxiliary run from the reach-deterministic state q is deterministic. When the auxiliary run reaches the accepting sink state in $\texttt{reach}(\mathcal{A})$, the run produced by the resolver which was simulating this auxiliary run in $\texttt{reach}(\mathcal{A})$ must also contain a significant transition. When the resolver's run takes a significant transition to a state p', it again starts simulating an auxiliary run from a good state q' that is language-equivalent to p' in $\mathcal{A}$. ◀

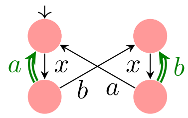
3 Graveyard of conjectures and the automata that killed them

We begin by analysing why naïve determinisation techniques fail. We provide increasingly complex counterexamples to specific "straw man" determinisation techniques, which then help us build the intuition required for our main result. These examples demonstrate the pitfalls that our final construction must avoid. Although the definition of the 65-state automaton can be understood directly by skipping ahead to Section 4, we strongly recommend reading this section first to understand the method behind the madness of the final construction.

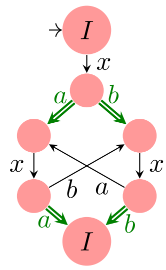
A motivating example. We start with the example automaton $\mathcal{A}_{\mathsf{BKKS}}$ from Figure 1. We show two deterministic Büchi automata that are language-equivalent to $\mathcal{A}_{\mathsf{BKKS}}$ and that have at most as many states as $\mathcal{A}_{\mathsf{BKKS}}$ in Figure 3.

First, observe that the 4 bottom states of $\mathcal{A}_{\mathsf{BKKS}}$ are reach-deterministic, while the non-reach-deterministic states are $\{I, \iota_a, \iota_b\}$. We refer to these 3 states as the 'nondeterministic component' of $\mathcal{A}_{\mathsf{BKKS}}$.

The automaton on the left of Figure 3 is obtained by *rewiring*: keeping the deterministic component and redirecting significat transitions. This is formally defined in Section 3.1. The second automaton, on the other hand, is obtained by replacing the 'nondeterministic component' of $\mathcal{A}$ with a deterministic part with no more states.



(a) Rewiring



(b) Replacing the nondeterministic component

Figure 3 Two language-equivalent deterministic Büchi automata to the automaton from Figure 1.

We describe automata where these two determinisation techniques do not work (without strictly increasing the number of states) in Sections 3.1 and 3.3, respectively. In Section 3.2 we present yet another determinisation tactic and explain how to defeat it. We then use these examples and insights to motivate our succinct HD Büchi automaton in Section 4.

3.1 The Rewiring Conjectures

For a simplified Büchi automaton $\mathcal{A}$, a *rewiring* of $\mathcal{A}$ is a deterministic Büchi automaton $\mathcal{B}$ that satisfies the following two conditions.

1. The states of $\mathcal{B}$ are the reach-deterministic states of $\mathcal{A}$, and the non-significant transitions of $\mathcal{B}$ are the same as the non-significant transitions that are outgoing from the reach-deterministic states of $\mathcal{A}$.
2. The significant transitions of $\mathcal{B}$ are of the form $p \overset{a}{\Rightarrow} q$, such that $L(\mathcal{A}, q) = a^{-1}L(\mathcal{A}, p)$.

In other words, $\mathcal{B}$ is obtained by redirecting the significant transitions from reach-deterministic states of $\mathcal{A}$ to other reach-deterministic states of $\mathcal{A}$, so that only reach-deterministic states are reached. In particular, $\mathcal{B}$ has no more states than $\mathcal{A}$. Prakash [Pra25, Conjecture 2] thus proposed what we call the *weak-rewiring conjecture*.

▶ **Conjecture 5** (Weak-rewiring conjecture). *Every simplified HD Büchi automaton has a language-equivalent rewiring.*

Towards disproving the weak-rewiring conjecture, we first disprove a stronger version of the conjecture. The strong-rewiring conjecture states that we can redirect significant transitions towards reach-deterministic states, while preserving the language, one-by-one and in an arbitrary order. Whereas in the weaker conjecture, we are allowed to redirect all significant transitions at once, which may be necessary to preserve language-equivalence.

▶ **Conjecture 6** (Strong-rewiring conjecture). *Let $\mathcal{A}$ be a simplified HD Büchi automaton, and let p be a reach-deterministic state in $\mathcal{A}$ such that there is transition of the form $p \xRightarrow{a} p'$, where p' is not a reach-deterministic state. Then there exists a reach-deterministic state q in $\mathcal{A}$ such that $L(\mathcal{A}, q) = a^{-1}L(\mathcal{A}, p)$, and the automaton $\mathcal{B}$, obtained by removing all transitions of the form $p \xrightarrow{a} p''$ in $\mathcal{A}$ and adding the transition $p \xRightarrow{a} q$, is language-equivalent to $\mathcal{A}$.*

3.1.1 Strong rewiring ☠

To disprove the strong-rewiring conjecture, we will consider the automaton $\mathcal{A}_{\texttt{strong}}$ in Figure 4, over the alphabet $\Sigma = \{a, b, c, x, y\}$.

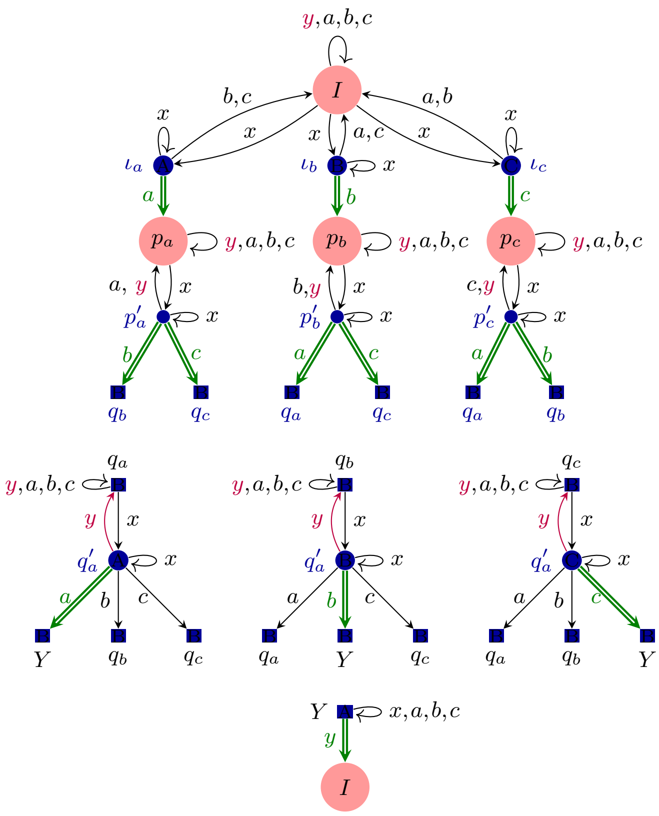


Figure 4 HD Büchi automaton $\mathcal{A}_{\texttt{strong}}$ for which the strong-rewiring conjecture is false. To improve readability, we represent the automaton "by chunks"; note that some states appear multiple times (in total, the automaton has 17 states).

We describe the language recognised by $\mathcal{A}_{\texttt{strong}}$. Define $L_\alpha = (x^*y^*)^*x\alpha$, for each $\alpha \in \{a, b, c\}$, and let

$$L_{\alpha\beta\beta} = L_\alpha L_\beta L_\beta.$$

The language recognised by $\mathcal{A}_{\texttt{strong}}$ in Figure 4 consists of words containing infinitely often infixes in $L_{\alpha\beta\beta}$, for some $\alpha \neq \beta$, as well as containing infinitely many ys. Formally, the language $L(\mathcal{A}_{\texttt{strong}})$ is the following:

$$L_{\texttt{strong}} = (\Sigma^* \cdot \bigcup_{\alpha\neq\beta\in\{a,b,c\}} L_{\alpha\beta\beta} \cdot \Sigma^* \cdot y)^\omega.$$

To intuitively see that this automaton indeed recognises the language $L_{\texttt{strong}}$, note that whenever an infix $L_\alpha L_\beta L_\beta$ is read from I, with $\alpha \neq \beta$, the automaton can produce a run

$$I \xrightarrow{L_\alpha} p_\alpha \xrightarrow{L_\beta} q_\beta \xrightarrow{L_\beta} Y.$$

From Y, the automaton takes a significant transition and moves to I upon reading letter y.

Although not necessary for establishing Theorem 2, we provide full proofs of the next result, as this is a similar, but simpler, version of the proof of correctness for our main automaton in Section 4.

▶ **Lemma 7.** *The automaton $\mathcal{A}_{\texttt{strong}}$ recognises the language $L_{\texttt{strong}}$. Moreover, it is history-deterministic and simplified.*

Proof. For the second claim, we need to show that $\mathcal{A}_{\texttt{strong}}$ is semantically deterministic and has reach-covering (which implies HDness by Lemma 4).

$L(\mathcal{A}_{\texttt{strong}}) = L_{\texttt{strong}}$ *and semantic determinism.* First, we observe that $L_{\texttt{strong}}$ is *prefix-independent*, i.e., for every letter $\sigma \in \Sigma$, $\sigma^{-1}L_{\texttt{strong}} = L_{\texttt{strong}}$. We will next show that for all states q in $\mathcal{A}_{\texttt{strong}}$, $L_{\texttt{strong}} \subseteq L(\mathcal{A}_{\texttt{strong}}, q)$.

Let w be a word in $L_{\texttt{strong}}$. We build an accepting run from q on w. Since w contains infinitely many y's, an arbitrary run from q eventually reaches one of the states $I, p_a, p_b, p_c, q_a, q_b, q_c$ (after reading a y). Furthermore, we observe the following language-equalities:

$$L(\texttt{reach}(\mathcal{A}_{\texttt{strong}}), I) = L_a \cup L_b \cup L_c,$$
$$L(\texttt{reach}(\mathcal{A}_{\texttt{strong}}), p_\alpha) = \bigcup_{\beta\neq\alpha} L_\beta,$$
$$L(\texttt{reach}(\mathcal{A}_{\texttt{strong}}), q_\alpha) = L_\alpha \ \cup \ \bigcup_{\beta\neq\alpha} L_\beta L_\beta.$$

Due to the above language-equations, every word w in $L_{\texttt{strong}}$ and every state q in the set $\{I, p_a, p_b, p_c, q_a, q_b, q_c\}$ is such that some prefix of w is in $L(\texttt{reach}(\mathcal{A}_{\texttt{strong}}), q)$. Thus, there is a finite run from q on w in $\mathcal{A}_{\texttt{strong}}$ that contains a significant transition. It follows by induction and the prefix independence of $L_{\texttt{strong}}$ that there is a run containing infinitely many significant transitions from every state of $\mathcal{A}_{\texttt{strong}}$ on every word in $L_{\texttt{strong}}$, as desired.

For the other language-inclusion $L(\mathcal{A}_{\texttt{strong}}, q) \subseteq L_{\texttt{strong}}$ for every state q, observe that every accepting run in $\mathcal{A}_{\texttt{strong}}$ must take the transition $Y \overset{y}{\Rightarrow} I$ infinitely often. Let L be the set of finite words on which there is a run ρ that starts at I and ends with the transition $Y \xrightarrow{y} I$. Then,

$$L = \Sigma^* \cdot \bigcup_{\alpha\neq\beta\in\{a,b,c\}} L_{\alpha\beta\beta} \cdot \Sigma^* \cdot y.$$

Thus, $L(\mathcal{A}_{\texttt{strong}}, q) \subseteq L^\omega = L_{\texttt{strong}}$, as desired.

$\mathcal{A}_{\texttt{strong}}$ *has reach covering.* Observe that every state in $\mathcal{A}_{\texttt{strong}}$ apart from $I, \iota_a, \iota_b, \iota_c$ is reach-deterministic. We show that

1. I simulates q_a in $\texttt{reach}(\mathcal{A}_{\texttt{strong}})$, and
2. ι_α simulates q'_α in $\texttt{reach}(\mathcal{A}_{\texttt{strong}})$, for each $\alpha \in \{a, b, c\}$.

This implies reach covering for $\mathcal{A}_{\texttt{strong}}$.

To see this, note that the only nondeterministic choice appears on state I, over the letter x. Eve has the following strategy to win the simulation-game: If Adam's token is in q_α and Adam gives the letter x, then Eve takes the transition on x to ι_α. If Eve uses this strategy and the simulation game in $\texttt{reach}(\mathcal{A}_{\texttt{strong}})$ starts from the starting positions (q_a, I) or $(q'_\alpha, \iota_\alpha)$, then

1. whenever Eve's state is in I, Adam's state is q_α for some α in $\{a, b, c\}$, and
2. Eve's state is in ι_α if and only if Adam's state is in q'_α.

In particular, the only way for Adam's token to reach the accepting sink state in $\mathcal{A}_{\texttt{reach}}$ is to read α from q'_α, but then Eve's token at ι_α in the simulation game would also reach the accepting sink state. Eve's strategy is therefore winning in the simulation games in $\texttt{reach}(\mathcal{A}_{\texttt{strong}})$, and it follows that $\mathcal{A}_{\texttt{strong}}$ has reach-covering. We conclude that $\mathcal{A}_{\texttt{strong}}$ is simplified, and therefore, due to Lemma 4, $\mathcal{A}_{\texttt{strong}}$ is HD. ◀

The following lemma then disproves the strong-rewiring conjecture.

▶ **Lemma 8.** *Let s be a reach-deterministic state in $\mathcal{A}_{\texttt{strong}}$. Let $\mathcal{B}$ be the automaton obtained by replacing the transition $Y \xrightarrow{y} I$ in $\mathcal{A}_{\texttt{strong}}$ by the transition $Y \xRightarrow{y} s$. Then,*

$$L(\mathcal{B}) \neq L(\mathcal{A}_{\texttt{strong}}).$$

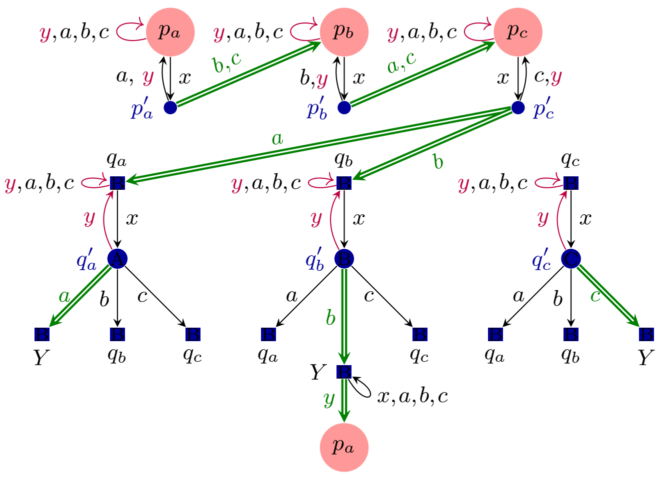
Proof. We show that there is word accepted by $\mathcal{B}$ that is rejected by $\mathcal{A}_{\texttt{strong}}$. We have the following three cases for the state s:

Case 1. $s \in Y$. Then y^ω is accepted by $\mathcal{B}$, but is not in $L_{\texttt{strong}}$.

Case 2. $s = p_\alpha$ or $s = p'_\alpha$. Then $\mathcal{B}$ accepts the word $(yx\beta x\beta y)^\omega$ that is not in $L_{\texttt{strong}}$, where $\beta \neq \alpha$ is in $\{a, b, c\}$.

Case 3. $s = q_\alpha$ or q'_α. Then, $(x\alpha y)^\omega$ is accepted by $\mathcal{B}$ but is not in $L_{\texttt{strong}}$ ◀.

Lemmas 7 and 8 together imply that the strong-rewiring conjecture is false. However, we observe that our counterexample for the strong-rewiring conjecture is not a counterexample for the weak-rewiring conjecture. We present a language-equivalent rewiring $\mathcal{D}_{\texttt{strong}}$ in Figure 5.



■ **Figure 5** A language-equivalent rewiring $\mathcal{D}_{\texttt{strong}}$ for $\mathcal{A}_{\texttt{strong}}$.

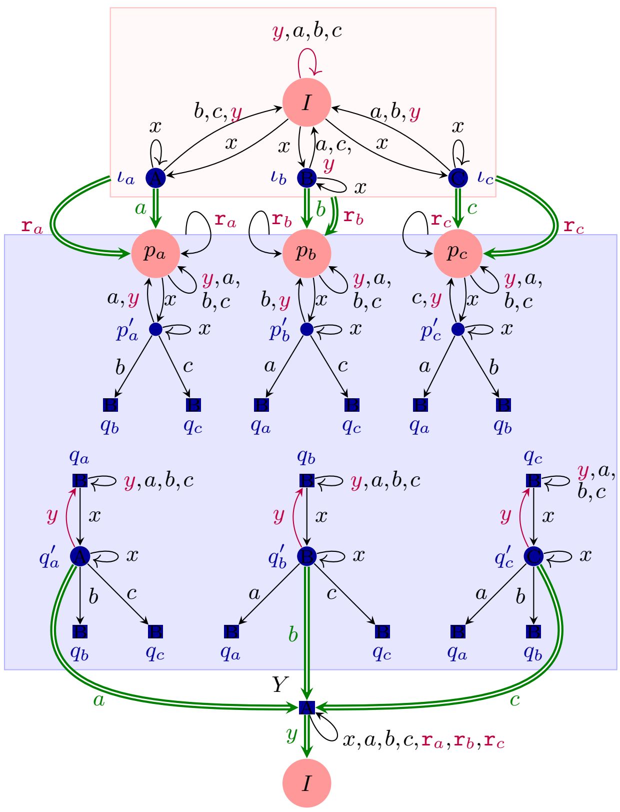
▶ **Lemma 9.** *The automaton $\mathcal{D}_{\texttt{strong}}$ is a language-equivalent rewiring of $\mathcal{A}_{\texttt{strong}}$.*

The proof of Lemma 9 is in Appendix A. We will next build upon $\mathcal{A}_{\texttt{strong}}$ to disprove the weak-rewiring conjecture (Conjecture 5).

3.1.2 Weak rewiring ☠

We now describe a counterexample to the weak-rewiring conjecture, which is the automaton $\mathcal{A}_{\texttt{weak}}$ shown in Figure 6. The automaton $\mathcal{A}_{\texttt{weak}}$ is obtained by modifying $\mathcal{A}_{\texttt{strong}}$ as follows.

1. The significant transitions from p's to qs are made non-significant.
2. For these transition to remain non-significant when the automaton is made normal, we add three new *reset letters* $\mathtt{r}_a, \mathtt{r}_b$, and $\mathtt{r}_c$. The transitions in $\mathcal{A}_{\texttt{weak}}$ over these reset letters look like the following:
   a. From every state s in $\{\iota, \iota_a, \iota_b, \iota_c\}$ we have a significant transition $s \xRightarrow{\mathtt{r}_\alpha} p_\alpha$ for each $\alpha \in \{a, b, c\}$.
   b. From state Y, we have non-significant self-loops on the reset letters.
   c. From every other state s, we have a non-significant transition $s \xrightarrow{\mathtt{r}_\alpha} p_i$, for each $\alpha \in \{a, b, c\}$.



■ **Figure 6** HD Büchi automaton $\mathcal{A}_{\texttt{weak}}$ for which the weak rewiring conjecture is not true. The transitions $\xrightarrow{\mathtt{r}_i} p_i$ outgoing from a box indicates that from every state in that box, there is a transition on $\mathtt{r}_i$ to p_i.

Changing transitions from the states p's to qs to be non-significant and adding the reset letters to obtain $\mathcal{A}_{\texttt{weak}}$ from $\mathcal{A}_{\texttt{strong}}$ has the effect of 'gluing' the reach-deterministic states apart from Y in a single strongly connected component in the automaton $\texttt{reach}(\mathcal{A}_{\texttt{weak}})$ (blue box in Fig. 6). This eliminates the possibility of a language-equivalent rewiring similar to $\mathcal{D}_{\texttt{strong}}$ in Figure 5, as we will prove in Lemma 10.

We begin by describing the language of $\mathcal{A}_{\texttt{weak}}$. Let $\Sigma' = \{x, a, b, c, y, \texttt{r}_a, \texttt{r}_b, \texttt{r}_c\}$ and

$$L'_{\alpha\beta\beta} = (L_\alpha + \texttt{r}_\alpha)L_\beta L_\beta \ , \quad \text{for } \alpha, \beta \in \{a, b, c\},$$

where $L_\alpha = (x^*y^*)^*x\alpha$ as previously. Then,

$$L_{\texttt{weak}} = ((x + a + b + c + y)^* \cdot \bigcup_{\alpha \neq \beta \in \{a,b,c\}} L'_{\alpha\beta\beta} \cdot \Sigma^* \cdot y)^\omega.$$

▶ **Lemma 10.** *The weak-rewiring conjecture is not true for $\mathcal{A}_{\texttt{weak}}$.*

Proof sketch. The proof that $\mathcal{A}_{\texttt{weak}}$ is simplified and recognises $L_{\texttt{weak}}$ is similar to that for $\mathcal{A}_{\texttt{strong}}$. We next show that there is no language-equivalent rewiring for $\mathcal{A}_{\texttt{weak}}$.

Let $\mathcal{B}$ be a rewiring of $\mathcal{A}_{\texttt{weak}}$. We observe that if $\mathcal{B}$ has a significant transition of the form $q'_\alpha \xRightarrow{\alpha} s$ for some $\alpha \in \{a, b, c\}, s \neq Y$, then $\mathcal{B}$ accepts the word $(\texttt{r}_a xbx\alpha x\alpha)^\omega$, which is not in $L_{\texttt{weak}}$ because w does not contain infinitely many y. Thus, in this case, $\mathcal{B}$ is not language-equivalent to $\mathcal{A}_{\texttt{weak}}$. We therefore assume that all significant transitions outgoing from states q'_a, q'_b, and q'_c are towards Y.

Suppose $\mathcal{B}$ has the transition $Y \xRightarrow{y} s$ for some state s in $\mathcal{B}$. The proof that $\mathcal{B}$ then accepts a word that is not recognised by $\mathcal{A}_{\texttt{weak}}$ is verbatim to the proof of Lemma 8. This disproves the weak-rewiring conjecture. ◀

3.2 Copying reach-deterministic states

The previous automaton $\mathcal{A}_{\texttt{weak}}$ may momentarily fill an optimistic researcher with the hope of having found an HD Büchi automaton strictly smaller than every language-equivalent deterministic one. Unfortunately, there are smaller deterministic automata recognising $L_{\texttt{weak}}$.

In Figure 7, we present a deterministic automaton $\mathcal{D}_{\texttt{weak}}$ recognising $L_{\texttt{weak}}$. This automaton is obtained by removing the non-reach-deterministic states $\{I, \iota_a, \iota_b, \iota_c\}$ from $\mathcal{A}_{\texttt{weak}}$ and adding two copies of the state Y, to have Y_a, Y_b, and Y_c.

This determinisation strategy works here because the number of Y-copies (2 in this case) is smaller than the number of non-reach-deterministic states (4 in this case). To defeat it, it suffices to add many copies of the gadgets, so that we would require to introduce > 4 copies of the state Y.

3.3 Determinising the nondeterministic component

We discuss a final determinisation technique. Recall the automaton $\mathcal{A}_{\texttt{BKKS}}$ in Figure 1. In Figure 3b, we showed a language-equivalent deterministic Büchi automaton of the same size that is not obtained by a rewiring, but instead by '*determinising the nondeterministic component*'. It is easy to come up with a similar language-equivalent deterministic Büchi automata for both $\mathcal{A}_{\texttt{weak}}$ and $\mathcal{A}_{\texttt{strong}}$, where we replace the states $I, \iota_a, \iota_b, \iota_c$ that are not reach-deterministic with two states as in Figure 3b, and add appropriate transitions. This provides yet a different proof of non-succinctness for $\mathcal{A}_{\texttt{weak}}$.

To defeat such a determinisation, we propose the automaton $\mathcal{A}_{\texttt{replace}}$ in Figure 8. This example is inspired by a recent example which arose in a different context [HPT25, Figure 6]. The automaton $\mathcal{A}_{\texttt{replace}}$ recognises the language

$$((L_1 + L_2)^*(L_1L_1 + L_2L_2))^\omega,$$

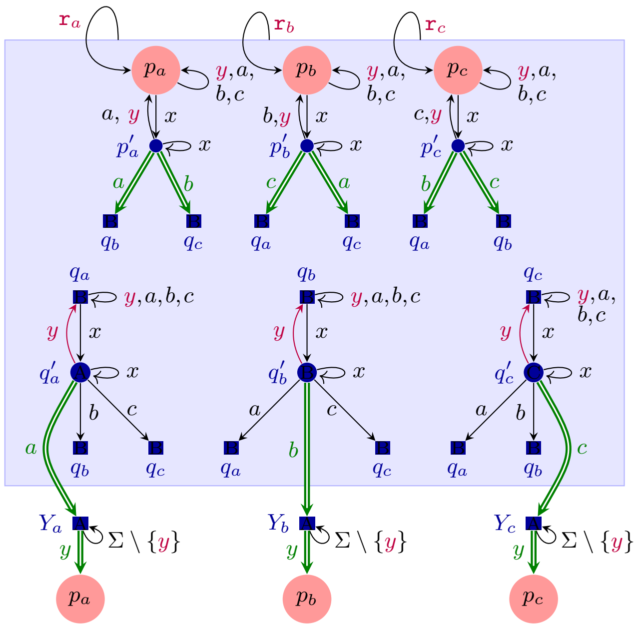


Figure 7 Determinisation of $\mathcal{A}_{\texttt{weak}}$ by introducing copies of the state Y.

for some appropriate L_1 and L_2 that are languages over finite words. For comparison, we remark that the language of $\mathcal{A}_{\texttt{BKKS}}$ is $(xa + xb)^*(xaxa + xbxb)^\omega$.

To describe L_1 and L_2 concretely, let $A = \{a, b, c\}$, and $\Sigma = \{a, b, c, 1, 2\}$. Define the languages $L_1' = \Sigma^* c1$, and $L_2' = \Sigma^* a\dot{A}^* b$. Then, L_1 is given by

$$L_1 = L_1' \setminus (L_2'\Sigma^*),$$

i.e., the set of words in L_1' with no prefix in L_2', and analogously, L_2 is defined as

$$L_2 = L_2' \setminus (L_1'\Sigma^*).$$

The only nondeterminism in $\mathcal{A}_{\texttt{replace}}$ is on I over the letter a. On this state, we 'guess' whether we are going to read a word in L_1 or L_2. The fact that we can use an HD resolver that eventually makes the right choice is proved similarly to the case of $\mathcal{A}_{\texttt{BKKS}}$ (Example 1).

Determinising the nondeterministic component of $\mathcal{A}_{\texttt{replace}}$, given by the states I, ι_a, ι_b and ι_c, requires 5 states (copying one of the 5-states components below). Therefore, such a determinisation strategy yields a deterministic automaton with strictly more states than $\mathcal{A}_{\texttt{replace}}$.

There is, of course, a language-equivalent rewiring for $\mathcal{A}_{\texttt{replace}}$, which is similar in spirit to the language-equivalent rewiring in Figure 3a for $\mathcal{A}_{\texttt{BKKS}}$. This provides a smaller language-equivalent deterministic automaton.

We are ready to propose an HD Büchi automaton strictly smaller than every language-equivalent deterministic one. In the next section, we combine our examples to construct the succinct HD Büchi automaton that defeats all these determinisation techniques at the same time.

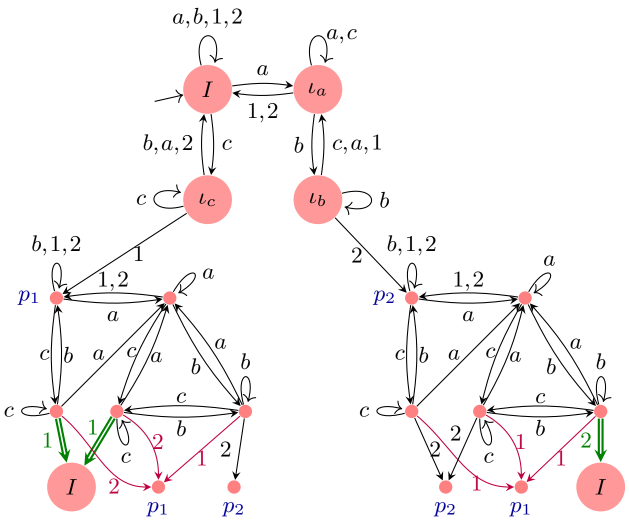


Figure 8 The HD Büchi automaton $\mathcal{A}_{\texttt{replace}}$.

4 A succinct HD Büchi automaton

In this section, we introduce the protagonist of our succinctness result: a 65-state Büchi automaton $\mathcal{A}_{\texttt{main}}$, and prove that it is in fact history deterministic. In Section 5, we will prove that there is no language-equivalent deterministic automaton with at most 65 states.

Some intuition. To build the automaton $\mathcal{A}_{\texttt{main}}$, we combine the ideas and gadgets explained in the previous section. In particular, a succinct HD Büchi automaton must satisfy the following (informal) properties,

No rewiring. It does not admit a language-equivalent rewiring.

No copying. A determinisation strategy based on making copies of a Y-state requires making more copies than states in the nondeterministic part.

No replacement. The nondeterministic part does not admit a determinisation with fewer states.

We ensure the no-rewiring property by using the "skeleton structure" of the automaton $\mathcal{A}_{\texttt{weak}}$ from Figure 6. That is, the general structure of the automaton $\mathcal{A}_{\texttt{main}}$ consists of

- a nondeterministic initial component with 4 states,
- several deterministic "basic blocks", each of them recognising a language of finite words L_i,
- a state Y checking for appearances of the letter y.

Moreover, we include reset letters $\mathtt{r}_i$ as in $\mathcal{A}_{\texttt{weak}}$ to 'glue together' the basic blocks in a single SCC of $\texttt{reach}(\mathcal{A}_{\texttt{main}})$.

We ensure the no-copying property by having $12 = 2 \cdot 6$ "basic blocks" in the skeleton structure. A determinisation via making copies of a Y-state would incur in adding 5 states, more than what we gain from removing the nondeterministic part.

To ensure the no-replacement property, we use the gadget from Figure 8. Each basic block, therefore, will be a subautomaton of 5 states similar to the one at the bottom left of Figure 8.

The language. Before introducing the automaton, we describe the language it recognises. We fix the alphabet

$$\Sigma = \{a, b, c, 1, 2, 3, 4, 5, 6, \mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3, \mathtt{r}_4, \mathtt{r}_5, \mathtt{r}_6, y\}.$$

For notational convenience we let $E = \{1, 2, 3\}$, $F = \{4, 5, 6\}$ and $A = \{a, b, c\}$.

The language $L_{\mathsf{main}} = L(\mathcal{A}_{\mathtt{main}})$ is built similarly to the language of $\mathcal{A}_{\mathtt{strong}}$. The "building blocks" of the language are 6 languages of finite words, $L_1, \ldots, L_6$ described below. Then, we let:

$$L_{\alpha\beta\beta} = (L_\alpha + \mathtt{r}_\alpha)L_\beta L_\beta, \quad \text{for } \alpha \neq \beta, \quad \alpha, \beta \in \{1, \ldots, 6\}.$$

Then, the final language is:

$$L_{\mathsf{main}} = (\Sigma^* \cdot \bigcup_{\substack{\alpha \neq \beta \\ 1 \leq \alpha, \beta \leq 6}} L_{\alpha\beta\beta} \cdot \Sigma^* \cdot y)^\omega$$

That is, L_{main} consists of the words containing infinitely often infixes in some $L_{\alpha\beta\beta}$ as well as containing infinitely many y's.

It remains to describe the languages L_α for $\alpha = 1, \ldots, 6$. This is best done by providing the *DFA-classifier* $\mathcal{A}_{\mathtt{class}}$ from Figure 9. This is a DFA over Σ with multiple final states, in this case, the six states $\blacksquare\ell_1, \ldots, \blacksquare\ell_6$. Its initial state is p. The language L_α is the set of words for which the run over $\mathcal{A}_{\mathtt{class}}$ ends up in $\blacksquare\ell_\alpha$. We note that $\blacksquare\ell_\alpha$ is *not* a sink: words that admit a proper prefix ending in some $\blacksquare\ell_\alpha$ are rejected. By definition, the sets $L_1, \ldots, L_6$ are pairwise disjoint.

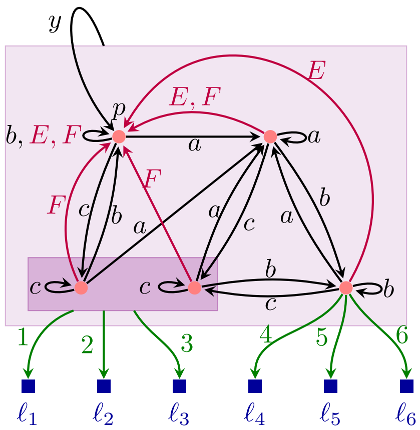


Figure 9 The DFA-classifier $\mathcal{A}_{\mathtt{class}}$ defining the languages $L_1, \ldots, L_6$.

We give some intuition for the DFA-classifier $\mathcal{A}_{\mathtt{class}}$. We first remark that $\mathcal{A}_{\mathtt{class}}$ contains no $\mathtt{r}_\alpha$ transitions: all words containing such a letter are rejected. Note that y acts as a reset letter: from all states, the transition on y goes back to state p. While no letter a is produced, the automaton stays in the two leftmost states; if the factor ce is read for some $e \in E = \{1, 2, 3\}$ is then read, it goes to some $\blacksquare\ell_e$. When letter a is read, we go to the right part of the automaton, and remain there while only letters in $A = \{a, b, c\}$ appear. From this right part:

- A^*ce make us advance to $\blacksquare\ell_e$ for $e \in E$, while A^*cF resets to p,

- A^*bf make us advance to $\blacksquare\ell_f$ for $f \in F$, while A^*bE resets to p, and
- $A^*a(E+F)$ resets to p.

The following claim follows from inspection of $\mathcal{A}_{\texttt{class}}$.

▷ Claim 11. Let w be finite word that contains an infix in a language L_α for some $\alpha \in [6]$, and no prefix in $L_{\alpha'}$ for $\alpha' \neq \alpha$. Then, either w is in L_α, or w contains a letter $\texttt{r}_\beta$ for some $\beta \in [6]$.

The automaton $\mathcal{A}_{\texttt{main}}$. We describe the HD Büchi automaton recognising L_{main}. The states of the automaton are:

- the initial state I, and states I_a, I_b and I_c;
- the "circular states": $\bullet p_i, \bullet q_i, \bullet r_i, \bullet s_i, \bullet t_i$ for $i \in \{1, \ldots, 6\}$;
- the "square states": $\blacksquare p_i, \blacksquare q_i, \blacksquare r_i, \blacksquare s_i, \blacksquare t_i$ for $i \in \{1, \ldots, 6\}$;
- and finally the state $\blacksquare Y$.

In total, it has $4 + 2 \times 5 \times 6 + 1 = 65$ states.

Visualising the automaton. The automaton is pictured in Figure 10. To visualise the 65-state automaton without creating a tangle of spaghetti, we split the transitions across two figures and use a few graphical shorthands similar to the earlier section. The transitions of the automaton over the letters a, b, c, and y are described in Figure 10a, whereas the transitions of the automaton over the rest of the letters $1, \ldots, 6$ and $\texttt{r}_1, \ldots, \texttt{r}_6$ are depicted in Figure 10b. Note that to maintain the planarity and readability of the diagrams, some states appear in multiple locations. We use variables to represent sets of transitions.

- The variable i ranges over the set $\{1, \ldots, 6\}$,
- the variable e ranges over the set $E = \{1, 2, 3\}$, and
- the variable f ranges over the set $F = \{4, 5, 6\}$.

Therefore, a single arrow to a state like $\bullet p_f$ in fact represents the transitions $\to \bullet p_4$, $\to \bullet p_5$, and $\to \bullet p_6$. Finally, the rectangles around states represent grouped sources. An arrow originating from a box around a set of states implies a transition from every state within that box. For instance, in Figure 10a, the arrow labelled y to the state $\bullet p_i$ indicates that on input y, the states $\bullet p_i, \bullet q_i, \bullet r_i, \bullet s_i$, and $\bullet t_i$ transition to state $\bullet p_i$.

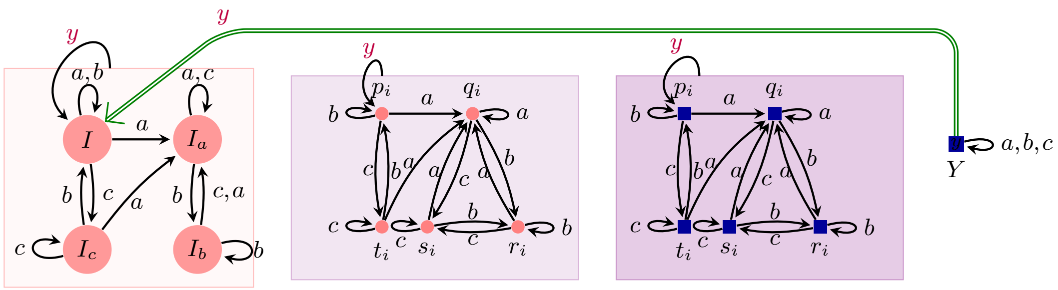
Working of the automaton. We note that there is a single nondeterministic choice: when reading letter a from state I (similar to the mechanism in Figure 8).

This automaton works as follows. It reads sequences of languages L_i until a factor $L_\beta L_\beta$ that has been preceded by some L_α or $\texttt{r}_\alpha$, with $\alpha \neq \beta$, appears. Then, it goes to state Y and waits for letter y to appear. When this happens, it restarts again.

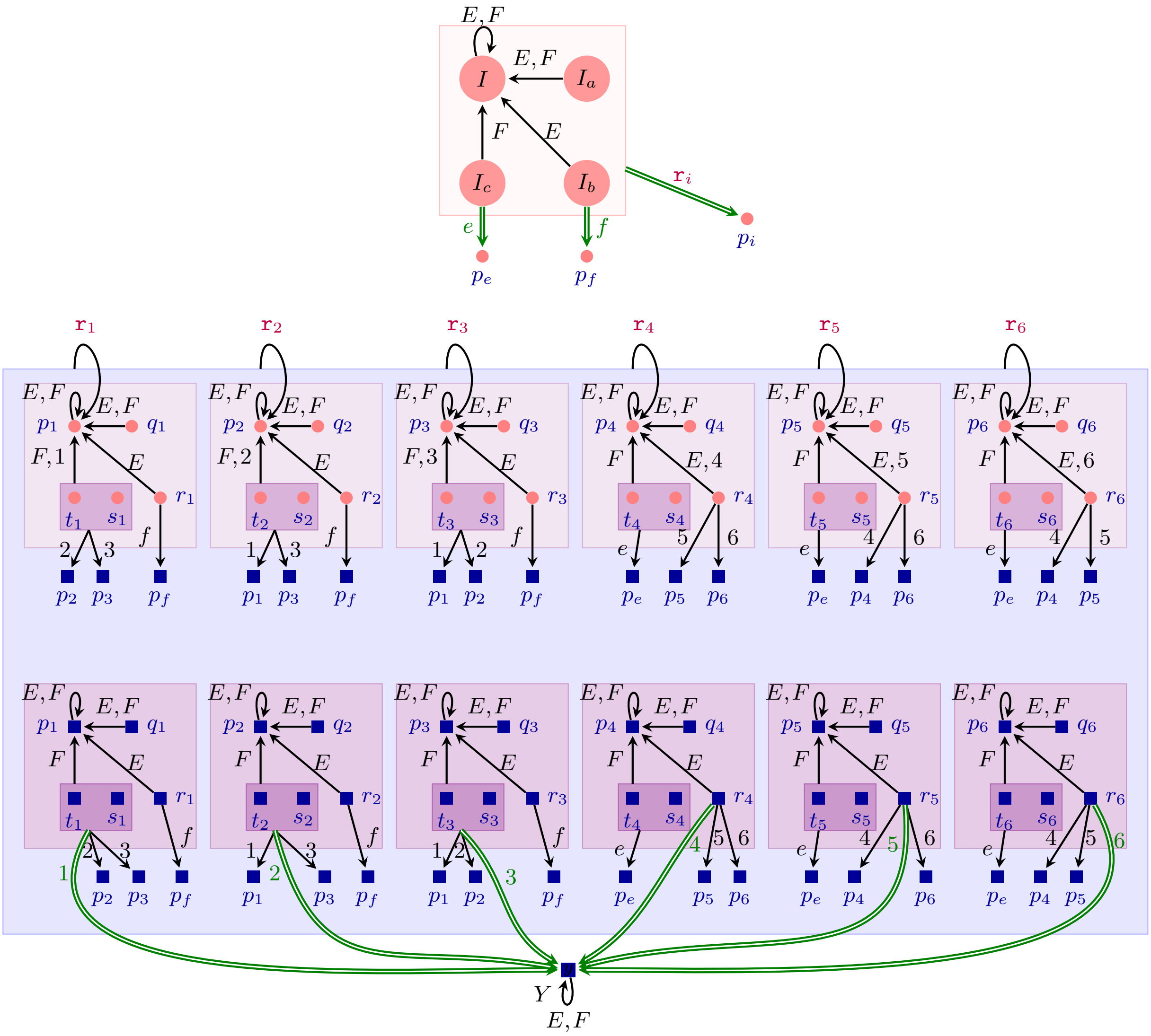
The top box (states I, I_a, I_b, I_c) waits for the first language L_α to appear (or letter $\texttt{r}_\alpha$). Using the nondeterministic choice on I over a, we will guess whether α is in E and we are going to see a word of the form A^*cE (we should wait in I), or whether α is in F and we are going to see a word of the form A^*aA^*bF.

If we make the right choice, we move to $\bullet p_\alpha$. If we make the wrong choice, no big deal! We go back to I, and since for word is in the language, some factor $L_\beta L_\beta$ must appear eventually. Therefore, we can use the information "$\alpha \in E$?" to resolve the nondeterminism the next time: if $\alpha \in E$, we wait in I (loop at state I on letter a), if $\alpha \in F$, we go to I_a whenever possible. Eventually, this strategy will make us advance to the big blue box over a word in L_{main}.

In the big blue box, the α^{th} box on the top row resets over L_α, and moves to the lower row over L_β if $\beta \neq \alpha$. More specifically, it goes to the state $\bullet p_\alpha$ over L_α and to $\blacksquare p_\beta$ over L_β.



(a) The automaton's transitions over the letters a, b, c, and y. The value of i ranges over the set $\{1, \ldots, 6\}$.



(b) The transitions over the letters $\{1, \ldots, 6\} \cup \{\mathtt{r}_1, \ldots, \mathtt{r}_6\}$. We let $E = \{1, 2, 3\}$ and $F = \{4, 5, 6\}$. The variables e and f ranges over the sets E and F, respectively.

Figure 10 The automaton $\mathcal{A}_{\mathtt{main}}$ whose transitions are depicted in two parts.

From $\blacksquare p_\beta$ in the lower row, it advances to Y over L_β. Over L_α, for $\alpha \neq \beta$, it stays in the lower row, going to $\blacksquare p_\alpha$. Over a letter $\mathtt{r}_i$ we reset to the upper row, to $\bullet p_i$; this ensures that the big blue box is strongly connected over non-significant transitions.

The automaton has three strongly connected components when restricted to non-significant transitions: the top pink box, the big blue box, and the state Y.

Correctness. The proof of the below result on $\mathcal{A}_{\mathtt{main}}$ is similar to the proof of Lemma 7. Full proofs are provided in Appendix B.

▶ **Theorem 12.** *The automaton $\mathcal{A}_{\mathtt{main}}$ is history-deterministic and recognises L_{main}. Moreover, it is simplified.*

5 There is no small deterministic automaton

In this Section, we prove that $\mathcal{A}_{\mathtt{main}}$ is succinct.

▶ **Theorem 13.** *Every deterministic Büchi automaton that is language-equivalent to $\mathcal{A}_{\mathtt{main}}$ has at least 66 states.*

To prove Theorem 13 above, we show the equivalent result that every deterministic coBüchi automaton recognising the complement language L^c_{main} has at least 66 states. Towards this, we rely on results from Abu Radi and Kupferman on the minimisation of HD coBüchi automata [AK22], which we describe in Section 5.1. We then present computer-aided lemmas about the size of minimal DFAs separating regular languages over finite words, an NP-complete problem [Gol78]. Using them, we conclude our proof in Section 5.3.

5.1 Structure of HD coBüchi automata

We briefly recall the results of Abu Radi and Kupferman's seminal paper [AK22] that showed a canonical form for HD coBüchi automata. We will then use these results to construct a state-minimal history-deterministic coBüchi automaton for the complement language of L_{main}. We begin by introducing a few notions that are necessary to state their results.

We say that a coBüchi automaton is a *safety automaton* if it has no significant transitions. For every coBüchi automaton $\mathcal{A}$, we use $\mathtt{safe}(\mathcal{A})$ to denote the safety automaton that is obtained by deleting all the significant transitions of $\mathcal{A}$.

A *safe component* of $\mathcal{A}$ is a strongly connected component of $\mathtt{safe}(\mathcal{A})$. We write $L_{\mathtt{safe}}(\mathcal{A}, q)$ to denote the language $L(\mathtt{safe}(\mathcal{A}), q)$, for a state q in $\mathcal{A}$, which we call the *safe language* of q.

We say that a coBüchi automaton is *normal* if every transition that is not in a safe component of $\mathcal{A}$ is a significant transition. Every coBüchi automaton can be made normal while preserving the acceptance of each run, by simply making significant every non-significant transitions of $\mathcal{A}$ that is not part of a safe-component. We assume that all states of automata are reachable from the initial state.

A coBüchi automaton $\mathcal{A}$ is *safe-deterministic* if the automaton $\mathtt{safe}(\mathcal{A})$ is deterministic. A coBüchi automaton $\mathcal{A}$ is *safe-minimal* if there are no two different states that are language-equivalent in $\mathcal{A}$ as well as in $\mathtt{safe}(\mathcal{A})$. It is *safe-centralised* if there are no two language-equivalent states p, q in different safe-components of $\mathcal{A}$ such that $L_{\mathtt{safe}}(\mathcal{A}, p) \subseteq L_{\mathtt{safe}}(\mathcal{A}, q)$.

Abu Radi and Kupferman proved the following result on minimality for HD coBüchi automata.

▶ **Lemma 14** ([AK22, Theorem 3.6]). *Let $\mathcal{A}$ be a history-deterministic coBüchi automaton that is normal, semantically deterministic, safe-deterministic, safe-minimal, and safe-centralised. Then, it is statewise minimal amongst all HD coBüchi automaton recognising $L(\mathcal{A})$.*

We also use the subsequent result from Abu Radi and Kupferman's work.

▶ **Lemma 15** ([AK22, Proposition 3.4]). *Let $\mathcal{A}$ and $\mathcal{B}$ be two language-equivalent history-deterministic coBüchi automata that are normal and safe-deterministic. Then for every state p in $\mathcal{A}$, there are states q in $\mathcal{A}$ and s in $\mathcal{B}$ such that $L(\mathcal{A}, p) = L(\mathcal{A}, q) = L(\mathcal{B}, s)$ and*

$$L_{\texttt{safe}}(\mathcal{A}, p) \subseteq L_{\texttt{safe}}(\mathcal{A}, q) = L_{\texttt{safe}}(\mathcal{B}, s).$$

5.2 Complementation of HD Büchi automata

In their 2024 paper, Abu Radi, Kupferman, and Leshkowitz describe a complementation procedure for HD Büchi automata that produces HD coBüchi automata with at most as many states [AKL24, Theorem 2]. This procedure builds an HD coBüchi automata for the complement language that has as set of states the reach-deterministic states of a simplified HD Büchi automata. If we use their procedure on our 65-state automaton $\mathcal{A}_{\texttt{main}}$, we obtain a 61-state history-deterministic coBüchi automaton $\mathcal{C}_{\texttt{main}}$. The states of the automaton $\mathcal{C}_{\texttt{main}}$ are the reach-deterministic states of $\mathcal{A}_{\texttt{main}}$. The non-significant transitions of $\mathcal{C}_{\texttt{main}}$ are the non-significant transitions outgoing from reach-deterministic states of $\mathcal{A}_{\texttt{main}}$. The significant transitions of $\mathcal{C}_{\texttt{main}}$ are transitions $p \xRightarrow{\sigma} q$ for every pair of states p, q in $\mathcal{C}_{\texttt{main}}$ and every letter σ. A rendering of the automaton is given in Figure 13 in Appendix C. It follows from Abu Radi, Kupferman, and Leshkowitz's work [AKL24, Proof of Theorem 9] that $L(\mathcal{C}_{\texttt{main}}) = L^c_{\mathsf{main}}$; we include a proof in the appendix for self-containment nevertheless.

▶ **Lemma 16.** *The automata $\mathcal{C}_{\texttt{main}}$ is history deterministic, and recognises the complement of the language recognised by $\mathcal{A}_{\texttt{main}}$.*

The proof of statewise-minimality for $\mathcal{C}_{\texttt{main}}$, stated next, is included in Appendix C.

▶ **Lemma 17.** *The automaton $\mathcal{C}_{\texttt{main}}$ is the statewise minimal HD coBüchi automaton recognising $L(\mathcal{C}_{\texttt{main}})$.*

Proof sketch. From Lemma 14, we need to prove that $\mathcal{C}_{\texttt{main}}$ is HD, semantically deterministic, safe-deterministic, normal, safe-minimal, and safe-centralised. By Lemma 16, $\mathcal{C}_{\texttt{main}}$ is HD. It is also SD because every pair of states has a transition from one to the other on every letter. The automaton $\mathcal{C}_{\texttt{main}}$ is safe-deterministic and normal by construction. Safe-centralisation follows easily, as there are only 2 safe-components. To show that $\mathcal{C}_{\texttt{main}}$ is safe-minimised, a case analysis is needed. ◀

We note that Lemma 17 also proves that every deterministic coBüchi automaton $\mathcal{D}$ recognising $L(\mathcal{C}_{\texttt{main}})$ requires at least 61 states. We thus need to prove that $\mathcal{D}$ additionally has at least 5 more states, which we do in the next subsection.

5.3 Proof of succinctness

We now prove the following result.

▶ **Lemma 18.** *Every deterministic coBüchi automaton that is language-equivalent to $\mathcal{C}_{\texttt{main}}$ has at least 66 states.*

Throughout this subsection, we fix $\mathcal{D}_{\texttt{main}}$ to be a deterministic coBüchi automaton that recognises $L(\mathcal{C}_{\texttt{main}})$. The proof is by contradiction: we assume that $\mathcal{D}_{\texttt{main}}$ has at most 65 states, and we will conclude that it cannot recognise $L(\mathcal{C}_{\texttt{main}})$.

We assume, without loss of generality, that $\mathcal{D}_{\texttt{main}}$ is normal, that is, all its non-significant transitions occur in some SCC in $\texttt{safe}(\mathcal{D}_{\texttt{main}})$. Note that because $L(\mathcal{C}_{\texttt{main}}) = L(\mathcal{D}_{\texttt{main}})$ is prefix-independent, every state of $\mathcal{D}_{\texttt{main}}$ recognises the language $L(\mathcal{D}_{\texttt{main}})$.

We denote the safe-component of $\mathcal{C}_{\texttt{main}}$ that consists of every state of $\mathcal{C}_{\texttt{main}}$ apart from Y by $\mathcal{S}^{\mathcal{C}}_{\textsf{safe}}$. We first prove existence of states in $\mathcal{D}_{\texttt{main}}$ whose safe-languages coincide with the states of $\mathcal{S}^{\mathcal{C}}_{\textsf{safe}}$ in $\mathcal{C}_{\texttt{main}}$ (Lemma 20). For this, we use Lemma 15.

▶ **Lemma 19.** *There is a state d_0 in $\mathcal{D}_{\texttt{main}}$ such that*

$$L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, d_0) = L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, \bullet p_1).$$

Proof. In Lemma 15, we let $\mathcal{C}_{\texttt{main}}$ be $\mathcal{A}$, $\mathcal{D}_{\texttt{main}}$ be $\mathcal{B}$ and let p be $\bullet p_1$. Then, there is a state q in $\mathcal{C}_{\texttt{main}}$ and a state r in $\mathcal{D}_{\texttt{main}}$ such that

$$L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, \bullet p_1) \subseteq L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, q) = L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, r).$$

Because $\mathcal{C}_{\texttt{main}}$ is safe-centralised (see proof of Lemma 17), $\bullet p_1$ and q are in the same safe-component of $\mathcal{C}_{\texttt{main}}$. Thus, let d_0 be such that we have the non-significant transition $r \xrightarrow{\texttt{r}_1} d_0$ in $\mathcal{D}_{\texttt{main}}$, which exists because $L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, q) = L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, r)$ and $q \xrightarrow{\texttt{r}_1} \bullet p_1$ is a non-significant transition in $\mathcal{C}_{\texttt{main}}$. It follows that the safe-language of d_0 in $\mathcal{D}_{\texttt{main}}$ is $L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, \bullet p_1)$, as desired. ◀

We let $\mathcal{S}^{\mathcal{D}}_{\textsf{safe}}$ be the safe-component of $\mathcal{D}_{\texttt{main}}$ that contains d_0. The following lemma follows from Lemma 19 above and the strong-connectivity of $\mathcal{S}^{\mathcal{C}}_{\textsf{safe}}$ and $\mathcal{S}^{\mathcal{D}}_{\textsf{safe}}$.

▶ **Lemma 20.** *For every state p in $\mathcal{S}^{\mathcal{C}}_{\textsf{safe}}$, there is a state q in $\mathcal{S}^{\mathcal{D}}_{\textsf{safe}}$ such that*

$$L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, q) = L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, p)$$

.

Since $\mathcal{C}_{\texttt{main}}$ is safe-minimal, this implies that $\mathcal{S}^{\mathcal{D}}_{\textsf{safe}}$ has at least 60 states. Similar to Lemma 19, we can also prove that there is a state d_y in $\mathcal{D}_{\texttt{main}}$ whose safe-language is equivalent to that of Y in $\mathcal{A}_{\texttt{main}}$.

▶ **Lemma 21.** *There is a state d_y in $\mathcal{D}_{\texttt{main}}$ such that*

$$L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, d_y) = L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, Y).$$

Proof. By Lemma 15, there is a state d_y in $\mathcal{D}_{\texttt{main}}$ and a state q in $\mathcal{C}_{\texttt{main}}$ such that

$$L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, Y) \subseteq L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, q) = L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, d_y).$$

We observe that q must be Y, since $\mathcal{C}_{\texttt{main}}$ is safe-centralised and the safe-component consisting of Y is a singleton. Thus, the above language inclusion is an equality, as desired. ◀

Thus, $\mathcal{D}_{\texttt{main}}$ has at least 61 states: at least 60 states in $\mathcal{S}^{\mathcal{D}}_{\textsf{safe}}$, one state d_y, and possibly more. Towards proving that $\mathcal{D}_{\texttt{main}}$ has at least 66 states, we need to introduce a few notions.

The subautomaton $\mathcal{D}_{\mathsf{core}}$. First, we consider the SCC decomposition of the safe-component $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ in the absence of transitions on y. We call this the y-SCC decomposition. For a state p in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$, we let $L_{y\mathtt{safe}}(p)$ be the language of words w such that w does not contain y, and the run on w from p stays in the same y-SCC of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$.

Let $\mathcal{D}_{\mathsf{core}}$ be a subautomaton that is an end-SCC in the y-SCC-decomposition of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$. We will prove that $\mathcal{D}_{\mathsf{core}}$ has at least 60 states.

▶ **Lemma 22.** *For every state q in $\mathcal{D}_{\mathsf{core}}$, there is a state p in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ such that*

$$L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q) = L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p).$$

Proof. Let d_0 be the state in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ given by Lemma 19, and $d_0 \xrightarrow{u} q$ a safe run in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ reaching q. Then, $\bullet p_1 \xrightarrow{u} p$ is the desired state. ◀

▶ **Lemma 23.** *For every state p in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$, the subautomaton $\mathcal{D}_{\mathsf{core}}$ contains a state q such that*

$$L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q) = L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p).$$

Proof. This follows from the observation that $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ is strongly-connected even in the absence of transitions on y.

Concretely, let q' be a state in $\mathcal{D}_{\mathsf{core}}$, and let p' in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ be a state such that

$$L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q') = L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p');$$

such a p' exists due to the previous lemma.

Let $p' \xrightarrow{u'} p$ be a safe run in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ such that u' does not contain letter y. Then the corresponding run from q' on u' in $\mathcal{D}_{\mathtt{main}}$ leads to the desired state q: note that q is in $\mathcal{D}_{\mathsf{core}}$ because $\mathcal{D}_{\mathsf{core}}$ is an end-y-SCC. ◀

Thus, we note that if the y-SCC decomposition of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ contains two end components, then the automaton $\mathcal{D}_{\mathtt{main}}$ has size at least 120. We thus suppose that the y-SCC decomposition of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ has exactly one end component, which we call $\mathcal{D}_{\mathsf{core}}$.

▶ **Lemma 24.** *Suppose that $\mathcal{D}_{\mathtt{main}}$ has at most 65 states. Then, the y-SCC decomposition of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ has exactly one end component, which we denote by $\mathcal{D}_{\mathsf{core}}$.*

$\mathcal{K}$-safety. Let $\mathcal{K}$ be the language of infinite words for which no prefix is in L_i for $i \in [6]$. This language $\mathcal{K}$ is recognised by the safety automaton shown in Figure 11. We let $\mathcal{K}_y$ be the language that recognises the subset of words in $\mathcal{K}$ that contain no y.

We now state the lemma for which we will provide a computer-assisted proof.

▶ **Lemma 25.** *Let $\mathcal{D}$ be a deterministic safety automaton that separates $\mathcal{K}$ (resp. $\mathcal{K}_y$) and $L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, q)$ for some state q in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$, i.e.,*

$$\mathcal{K} \subseteq L(\mathcal{D}) \subseteq L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, q).$$

Then, $\mathcal{D}$ has at least 5 states.

Finding minimal DFAs (or deterministic safety automata) separating regular languages is an NP-complete problem [Gol78]. We thus use DFAMiner, a SAT-solver based tool for separating regular languages [DLS24]. Nonetheless, instead of plugging a 60-state ($\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$) and a 5-state automaton (the automaton for $\mathcal{K}$) into this tool, we make certain reductions that allow us to considerably reduce the state-space of automata that we input. We describe precise details in Appendix D. We note two useful corollaries of Lemma 25 below.

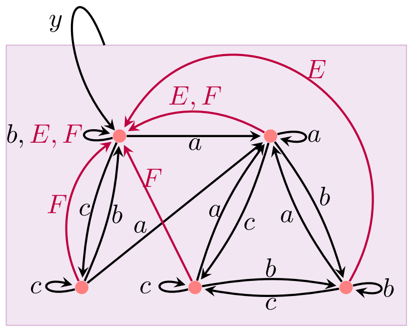


Figure 11 The automaton for the language $\mathcal{K}$. The initial state is the top left one. The automaton for language $\mathcal{K}_y$ is obtained by removing the transitions on y.

▶ **Corollary 26.** *Let q be a state of $\mathcal{D}_{\mathtt{main}}$ outside $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$. If*

$$\mathcal{K} \subseteq L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q),$$

then the safe-component containing q has at least 5 states.

Proof. From Lemma 15, there is a state p in $\mathcal{C}_{\mathtt{main}}$ such that $L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q) \subseteq L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p)$. This state p cannot be Y, since $\mathcal{K} \not\subseteq L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, Y)$. Thus, p is a state in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$. The conclusion then follows from Lemma 25. ◀

▶ **Corollary 27.** *Let q be a state of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ outside $\mathcal{D}_{\mathsf{core}}$. If*

$$\mathcal{K}_y \subseteq L_{y\mathtt{safe}}(q),$$

then the y-SCC containing q has at least 5 states.

Proof. Observe that $L_{y\mathtt{safe}}(q) \subseteq L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q)$. Since q is in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$, there is a state $p \in \mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ such that $L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q) = L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p)$. We thus obtain that

$$\mathcal{K}_y \subseteq L_{y\mathtt{safe}}(q) \subseteq L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p),$$

and the conclusion follows from Lemma 25. ◀

Observe that if there is a state q outside $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ with

$$\mathcal{K} \subseteq L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q),$$

then the safe-component of q is disjoint from the safe-component consisting of d_y, since $\mathcal{K} \not\subseteq L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, d_y)$ and $L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, d_y)$ is prefix-independent. Thus, in this case, $\mathcal{D}_{\mathtt{main}}$ will have at least 66 states: the 60 states of $\mathcal{D}_{\mathsf{core}}$, the state d_Y, and these additional 5 states that Corollary 26 implies the existence of.

Similarly, if there is a state q outside $\mathcal{D}_{\mathsf{core}}$ with $\mathcal{K}_y \subseteq L_{y\mathtt{safe}}(q)$ then Corollary 27 implies that $\mathcal{D}_{\mathtt{main}}$ has at least 66 states: the 60 states of $\mathcal{D}_{\mathsf{core}}$, the state d_Y, and these additional 5 states that Corollary 27 implies the existence of.

We thus obtain the following.

▶ **Lemma 28.** *If $\mathcal{D}_{\mathtt{main}}$ has at most 65 states, then there are no states q outside $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ with $\mathcal{K} \subseteq L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q)$ and there are no states p in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ outside $\mathcal{D}_{\mathsf{core}}$ with $\mathcal{K}_y \subseteq L_{y\mathtt{safe}}(p)$.*

We conclude our proof in the next lemma, whose full proof is given in Appendix C.

▶ **Lemma 29.** *If $\mathcal{D}_{\texttt{main}}$ has at most 65 states, then it does not recognise $L(\mathcal{C}_{\texttt{main}})$.*

Proof sketch. Let $\{o_1, \ldots, o_5\}$ be the (at most) 5 states outside $\mathcal{D}_{\textsf{core}}$ (note that d_y is one of these 5 states.) Using the previous lemma, for $i = 1, \ldots, 5$ we find words $o_i \xrightarrow{u_i} q_i$ such that u_i is a prefix of a word in $\mathcal{K}$ and q_i satisfies $L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, q_i) = L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, \bullet p_j)$, for some $j \in [6]$. Since there are only 5 states outside $\mathcal{D}_{\textsf{core}}$, there is some $j_0 \in [6]$ that is not covered by such a word. We show then that $\mathcal{D}_{\texttt{main}}$ rejects a word in

$$L = (\{u_1, \ldots, u_5\} L_{j_0}^2)^{\omega}.$$

But note that $L \subseteq L(\mathcal{C}_{\texttt{main}})$, because words in L do not contain any infix in $L_{\alpha\beta\beta}$. ◀

6 Conclusion and future directions

In this paper, we have solved the succinctness question for history-deterministic Büchi automata by exhibiting a 65-state HD Büchi automaton such that every language-equivalent deterministic Büchi automaton requires at least 66 states. The construction and the proof are involved and required computer assistance for some intermediate steps. In order to have a better understanding of these automata and make progress in further questions, a first step would be to find a simpler example of a succinct history-deterministic Büchi automaton. The existence of such automata is immediately unclear; even the smallest Büchi automaton that we know to be not determinisable by pruning has 7 states.

The immediate next question is to settle the width of the gap between deterministic and HD Büchi automata. More precisely: is there a family of HD Büchi automata such that language-equivalent deterministic Büchi automata need at least quadratic number of states? While our result kills the "no gap" conjecture (and rules out approaches like conjecture 5), constructing a family with a quadratic gap remains a challenge, especially given the complexity required just to establish a separation of one state.

Beyond theory, the techniques we developed to establish lower bounds on deterministic state complexity have practical potential. An interesting future direction would be to integrate these heuristics into SAT-based minimization tools like `SPOT` [DRC+22]. More precisely, given a deterministic Büchi automaton we can find a non-trivial lower bound on the size of every language-equivalent deterministic (or HD) automaton by minimising the coBüchi automaton for its complement language, following [AK22] and as described in Section 5.2. Can we further use these minimal HD coBüchi automata to obtain better minimisation algorithms for deterministic coBüchi automata?

References

AE25 Bader Abu Radi and Rüdiger Ehlers. Characterizing the polynomial-time minimizable ω-automata. *CoRR*, volume abs/2504.20553, 2025. doi: `http://doi.org/10.48550/ARXIV.2504.20553`.

AJP24 Rohan Acharya, Marcin Jurdziński, and Aditya Prakash. Lookahead games and efficient determinisation of history-deterministic Büchi automata. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 124:1–124:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024. doi: `http://doi.org/10.4230/LIPIcs.ICALP.2024.124`.

AK22 Bader Abu Radi and Orna Kupferman. Minimization and canonization of GFG transition-based automata. *Log. Methods Comput. Sci.*, volume 18(3), 2022. doi: http://doi.org/10.46298/lmcs-18(3:16)2022.

AKL24 Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz. Easy complementation of history-deterministic büchi automata. In *ATVA 2024*, volume 15054, pages 67–88. Springer, 2024.

BKKS13 Udi Boker, Denis Kuperberg, Orna Kupferman, and Michal Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100. Springer, 2013. doi: http://doi.org/10.1007/978-3-642-39212-2_11.

BL23 Udi Boker and Karoliina Lehtinen. When a little nondeterminism goes a long way: An introduction to history-determinism. *ACM SIGLOG News*, volume 10(1):24–51, 2023. doi: http://doi.org/10.1145/3584676.3584682.

CCFL24 Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. From Muller to parity and Rabin automata: Optimal transformations preserving (history) determinism. *TheoretiCS*, volume 3, 2024. doi: http://doi.org/10.46298/THEORETICS.24.12.

CCL22 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: http://doi.org/10.4230/LIPIcs.ICALP.2022.117.

CLW26 Antonio Casares, Christof Löding, and Igor Walukiewicz. Layered automata: A canonical model for automata over infinite words. 2026. doi: https://arxiv.org/abs/2601.15940.

Col09 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009. doi: http://doi.org/10.1007/978-3-642-02930-1_12.

Col12 Thomas Colcombet. Forms of determinism for automata (Invited talk). In *STACS*, volume 14 of *LIPIcs*, pages 1–23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. doi: http://doi.org/10.4230/LIPICS.STACS.2012.1.

DLS24 Daniele Dell'Erba, Yong Li, and Sven Schewe. DFAMiner: Mining minimal separating DFAs from labelled samples. In *FM*, volume 14934 of *Lecture Notes in Computer Science*, pages 48–66. Springer, 2024.

DRC$^+$22 Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. From Spot 2.0 to Spot 2.10: What's new? In *CAV*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2022.

Ehl25 Rüdiger Ehlers. Rerailing automata. *CoRR*, volume abs/2503.08438, 2025. doi: http://doi.org/10.48550/ARXIV.2503.08438.

EK24a Rüdiger Ehlers and Ayrat Khalimov. Fully generalized reactivity(1) synthesis. In *TACAS*, pages 83–102. Springer Nature Switzerland, 2024.

EK24b Rüdiger Ehlers and Ayrat Khalimov. A naturally-colored translation from LTL to parity and COCOA. *CoRR*, volume abs/2410.01021, 2024. doi: http://doi.org/10.48550/ARXIV.2410.01021.

EKV21 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: http://doi.org/10.4171/AUTOMATA-2/16.

ERS24 Javier Esparza, Rubén Rubio, and Salomon Sickert. Efficient normalization of linear temporal logic. *Journal of the ACM*, volume 71(2):1–42, 2024.

ES22 Rüdiger Ehlers and Sven Schewe. Natural colors of infinite words. In *FSTTCS*, volume 250 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: `http://doi.org/10.4230/LIPICS.FSTTCS.2022.36`.

Gol78 E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, volume 37(3):302–320, 1978. doi: `http://doi.org/10.1016/S0019-9958(78)90562-4`.

HKR02 Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair simulation. *Inf. Comput.*, volume 173(1):64–81, 2002. doi: `http://doi.org/10.1006/inco.2001.3085`.

HP06 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer Science Logic, CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006. doi: `http://doi.org/10.1007/11874683_26`.

HPT25 Thomas A. Henzinger, Aditya Prakash, and K. S. Thejaswini. Resolving nondeterminism with randomness. In *MFCS*, volume 345 of *LIPIcs*, pages 57:1–57:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

KS15 Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata. In *ICALP (2)*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015. doi: `http://doi.org/10.1007/978-3-662-47666-6_24`.

KSV96 Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. Relating word and tree automata. In *LICS*, pages 322–332. IEEE Computer Society, 1996. doi: `http://doi.org/10.1109/LICS.1996.561360`.

Kup18 Orna Kupferman. Automata theory and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 107–151. Springer International Publishing, 2018. doi: `http://doi.org/10.1007/978-3-319-10575-8_4`.

LP25 Karoliina Lehtinen and Aditya Prakash. The 2-token theorem: Recognising history-deterministic parity automata efficiently. In *STOC*, pages 1839–1850. ACM, 2025.

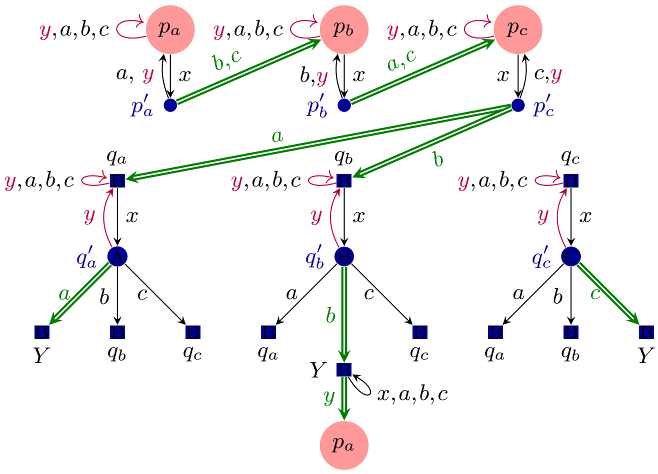
LW25 Christof Löding and Igor Walukiewicz. Minimal history-deterministic co-Büchi automata: Congruences and passive learning. In *Symposium on Logic in Computer Science (LICS)*, pages 431–443. 2025. doi: `http://doi.org/10.1109/LICS65433.2025.00039`.

Pra25 Aditya Prakash. History-deterministic parity automata : games, complexity, and the 2-token theorem, 2025. doi: `http://webcat.warwick.ac.uk/record=b4197352`. PhD thesis.

Sch10 Sven Schewe. Beyond hyper-minimisation—minimising dbas and dpas is np-complete. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi: `http://doi.org/10.4230/LIPIcs.FSTTCS.2010.400`.

A Proof of Lemma 9: Rewiring of $\mathcal{A}_{\texttt{strong}}$

▶ **Lemma 9.** *The automaton $\mathcal{D}_{\texttt{strong}}$ is a language-equivalent rewiring of $\mathcal{A}_{\texttt{strong}}$.*



■ **Figure 12** A language-equivalent rewiring $\mathcal{D}_{\texttt{strong}}$ for $\mathcal{A}_{\texttt{strong}}$. Redrawn from Figure 5.

Proof. The fact that $\mathcal{D}_{\texttt{strong}}$ is a rewiring of $\mathcal{A}_{\texttt{strong}}$ follows from the definition.

We can prove that every word in $L_{\texttt{strong}}$ is also in $L(\mathcal{D}_{\texttt{strong}})$ similar to how we proved $\mathcal{A}_{\texttt{strong}}$ is semantically deterministic in Lemma 7. Indeed, for every state q, $\texttt{reach}(\mathcal{D}_{\texttt{strong}}, q)$ and $\texttt{reach}(\mathcal{A}_{\texttt{strong}}, q)$ are structurally-isomorphic. For the other direction, consider the following languages:

$$L_{\neq} = \bigcup_{\alpha\neq\beta\in\{a,b,c\}} L_\alpha L_\beta \quad , \quad L_{=} = \bigcup_{\alpha\in\{a,b,c\}} L_\alpha L_\alpha,$$

and

$$L_{\texttt{Inf}(\neq)} = (\Sigma^* \cdot L_{\neq})^\omega \quad , \quad L_{\texttt{Inf}(=)} = (\Sigma^* L_{=})^\omega \quad , \quad L_{\texttt{Inf}(y)} = (\Sigma^* y)^\omega.$$

Then, note that

$$L_{\texttt{strong}} = L_{\texttt{Inf}(\neq)} \cap L_{\texttt{Inf}(=)} \cap L_{\texttt{Inf}(y)}.$$

We show that every word $w \in L(\mathcal{D}_{\texttt{strong}})$ is indeed in each of these three languages.

For two states s, s' in $\mathcal{D}_{\texttt{strong}}$, we let

$$L(s, s') = \{\text{Words that have a run from } s \text{ to } s'\}.$$

1. $L(\mathcal{D}_{\texttt{strong}}) \subseteq L_{\texttt{Inf}(\neq)}$. This is true because every accepting run in $\mathcal{D}_{\texttt{strong}}$ must visit p_a infinitely many times, as well as q_a or q_b infinitely many times. We observe that

$$L(p_a, q_a) \cup L(p_a, q_b) \subseteq \Sigma^* \cdot L_{\neq} \cdot \Sigma^*$$

2. $L(\mathcal{D}_{\texttt{strong}}) \subseteq L_{\texttt{Inf}(=)}$. Observe that this follows because

$$L(p'_c, Y) \subseteq \Sigma^* \cdot L_{=} \cdot \Sigma^*,$$

and every accepting run in $\mathcal{D}_{\texttt{strong}}$ must visit p'_c and Y infinitely often.

3. $L(\mathcal{D}_{\texttt{strong}}) \subseteq L_{\texttt{Inf}(y)}$. This follows because

$$L(Y, p_a) = \Sigma^* \cdot y \cdot \Sigma^*,$$

and every accepting run in $\mathcal{D}_{\texttt{strong}}$ must visit p_a and Y infinitely often.

We conclude that $L(\mathcal{D}_{\texttt{strong}}) = L(\mathcal{A}_{\texttt{strong}})$. ◀

B Proofs in Section 4: Correctness of $\mathcal{A}_{\texttt{main}}$

▶ **Theorem 12.** *The automaton $\mathcal{A}_{\texttt{main}}$ is history-deterministic and recognises L_{main}. Moreover, it is simplified.*

By Lemma 4, being a simplified Büchi automaton implies being history deterministic. Recall that a Büchi automaton is simplified if it is semantically deterministic, saturated, and has reach-covering.

▶ **Proposition 30.** *Every state in $\mathcal{A}_{\texttt{main}}$ recognises the language L_{main} and $\mathcal{A}_{\texttt{main}}$ is semantically deterministic.*

Proof. We remark that the language L_{main} is prefix-independent. Therefore, semantic determinism of $\mathcal{A}_{\texttt{main}}$ follows from the first part of the claim.

We first show that $L_{\mathsf{main}} \subseteq L(\mathcal{A}_{\texttt{main}}, q)$ for every state q. We will do this by inductively build an accepting run from q on an arbitrary word in L_{main}. At each step (except possibly the initialization step), we will make sure that a significant transition is taken. Note that all significant transitions of $\mathcal{A}_{\texttt{main}}$ end in some state in

$$Q_{\texttt{signif}} = \{I\} \cup \bigcup_{\alpha} \{\bullet p_\alpha\} \cup \{Y\}.$$

As L_{main} is prefix-independent, every suffix of a word in L_{main} also belongs to L_{main}. Therefore, it suffices to show that every word in L_{main} admits a run reaching a significant transition from every state $q \in Q_{\texttt{signif}}$, that is, $L_{\mathsf{main}} \subseteq L(\texttt{reach}(\mathcal{A}_{\texttt{main}}), q)$, for all $q \in Q_{\texttt{signif}}$.

Initialization step. We show that a prefix of an arbitrary run reaches some state in $Q_{\texttt{signif}}$. Note that every word $w \in L_{\mathsf{main}}$ contains infinitely many y's. Upon reading the letter y, every run reaches some state in

$$Q_{\texttt{signif}} \cup \bigcup_{\alpha} \{\blacksquare p_\alpha\}.$$

If the run reaches $Q_{\texttt{signif}}$ we are done. Assume that we reach a state $\blacksquare p_\alpha$. Let w' be the suffix of w from this point. Note that w' must contain an infix in $(L_{\alpha'} + \texttt{r}_{\alpha'})L_\beta$ for some distinct $\alpha', \beta \in [6]$: we consider the earliest occurrence of such an infix. Then, from Claim 11, w' either contains a reset letter or some prefix of w' is in $(L_{\alpha'}L_\beta$. If w' contains a reset letter, the run reaches a state in $Q_{\texttt{signif}}$ upon reading that letter. Otherwise, if $\alpha \neq \alpha'$, then the run reaches Y upon reading the prefix in $L_{\alpha'}$. If $\alpha = \alpha'$, then the run reaches Y upon reading the prefix in $L_\alpha L_\beta$.

Induction step. To build the run in the induction step, we distinguish three cases, according to the type of state. From Y, we see a significant transition upon reading y. From I, we note that $L(\texttt{reach}(\mathcal{A}_{\texttt{main}}), I)$ contains the words in $\bigcup_\alpha (L_\alpha + \Sigma^* \texttt{r}_\alpha)$. Since w contains an infix in some L_α, by Claim 11, the word w contains a prefix in this language.

The most challenging case is when $q = \bullet p_i$, for some i. Since $w \in L_{\mathsf{main}}$, it has a prefix of the form

$$u_0 u_\alpha u_\beta u'_\beta, \quad \text{with} \ \ u_\alpha \in (L_\alpha + \mathtt{r}_\alpha), \ u_\beta, u'_\beta \in L_\beta, \ \text{and} \ \alpha \neq \beta.$$

Consider an arbitrary run from $\bullet p_i$ over this word. We only need to show that some prefix of this run contains a significant transition. If there is no such significant transition seen, the run stays in the (deterministic) blue box. If $u_\alpha = \mathtt{r}_\alpha$, the run over $u_0 u_\alpha$ finishes in $\bullet p_\alpha$. Since $\alpha \neq \beta$, we have the following run

$$\bullet p_\alpha \xrightarrow{u_\beta} \blacksquare p_\beta \xrightarrow{u'_\beta} \Rightarrow .$$

We, therefore, focus on the case that $u_\alpha \in L_\alpha$. The last letter of u_α then must be α. After reading α, the automaton is necessarily in some state $\bullet p_\gamma$ or $\blacksquare p_\gamma$. From all these states, except from $\bullet p_\beta$, we reach a significant transition upon reading $u_\beta u'_\beta$. We argue that after reading u_α we cannot end up in $\bullet p_\beta$. Assume by contradiction that this is the case. Then, it must be the case that the two predecessors of $\bullet p_\beta$ in the run are in the box B_β on the top row, where

$$B_\beta = \{\bullet p_\beta, \bullet q_\beta, \bullet t_\beta, \bullet s_\beta, \bullet r_\beta\}.$$

We distinguish the cases $\alpha \in E$ and $\alpha \in F$. If $\alpha \in E$, then u_α ends by $c\alpha$. However, we have

$$B_\beta \xrightarrow{c} \{\bullet t_\beta, \bullet s_\beta\} \xrightarrow{\alpha} \bullet p_\alpha \neq \bullet p_\beta \ , \quad \text{a contradiction.}$$

If $\alpha \in F$, then u_α ends by $aA^*b\alpha$. However, we have

$$B_\beta \xrightarrow{aA^*} \{\bullet q_\beta, \bullet s_\beta, \bullet r_\beta\} \xrightarrow{b} \bullet r_\beta \xrightarrow{\alpha} \bullet p_\alpha \neq \bullet p_\beta, \quad \text{a contradiction}$$

For the converse inclusion $L(\mathcal{A}_{\mathtt{main}}, q) \subseteq L_{\mathsf{main}}$ for every state q, let w be a word in $L(\mathcal{A}_{\mathtt{main}}, q)$. Then w contains infinitely many y's, as every accepting run of $\mathcal{A}_{\mathtt{main}}$ must go through Y infinitely often. To show that w contains infinitely many infixes in $(L_\alpha + \mathtt{r}_\alpha) L_\beta L_\beta$, we observe the following facts. By "consecutive occurrence" we mean that no state of type $\bullet p_\gamma$ or $\blacksquare p_\gamma$ appears in between.

- If u labels a path $I \xrightarrow{u} \Rightarrow \bullet p_\alpha$, then $u \in (L_\alpha + \Sigma^* \mathtt{r}_\alpha)$.
- If u labels a path between two consecutive occurrences of $\bullet p_\alpha$ and $\blacksquare p_\beta$, then $u \in L_\beta$ and $\alpha \neq \beta$.
- If u labels a path between two consecutive occurrences of $\blacksquare p_\alpha$ and $\blacksquare p_\beta$, then $u \in L_\beta$ and $\alpha \neq \beta$.
- If u labels a path between two consecutive occurrences of $\blacksquare p_\alpha$ and $\bullet p_\beta$, then $u \in \Sigma^* \mathtt{r}_\beta$.
- If u labels a path between two consecutive occurrences of $\blacksquare p_\beta$ and Y, then $u \in L_\beta$.

Consider a significant transition ending in Y on an accepting run over w in $\mathcal{A}_{\mathtt{main}}$. Then, w contains an infix in $(L_\alpha + \mathtt{r}_\alpha) L_\beta L_\beta$ for $\alpha \neq \beta$. This easily follows using the above properties, by considering the 3 last appearances of p-states or the state I, and doing a distinction of six cases according to whether these states are $\blacksquare p$, $\bullet p$ or I. ◀

▷ Claim 31. $\mathcal{A}_{\mathtt{main}}$ is normal.

Proof. It suffices to observe that each rectangle in the image represents a strongly connected component induced by non-significant transitions. The rectangle containing the states I, I_a, I_b, and I_c forms one such strongly connected component, all the other states but Y forms another such strongly connected component, and finally the state Y alone also forms the final component. Every transition between these three components are significant. ◀

▶ **Proposition 32.** $\mathcal{A}_{\mathtt{main}}$ *has reach-covering.*

Proof. Since all states in $\mathcal{A}_{\mathtt{main}}$ are language-equivalent (by Claim 30), $\mathcal{A}_{\mathtt{main}}$ is SD. The only nondeterminism on $\mathcal{A}_{\mathtt{main}}$ is over the transition $I \xrightarrow{a}$. Therefore, the only states in $\mathcal{A}_{\mathtt{main}}$ that are not reach-deterministic are the 4 states in the top pink box: $B_{\mathtt{top}} = \{I, I_a, I_b, I_c\}$. We will show that I, I_a, I_b, and I_c simulate $\bullet p_1, \bullet q_1, \bullet r_1$, and $\bullet t_1$, respectively. The proof in the four cases is identical.

Consider the simulation game of $(\mathcal{A}_{\mathtt{main}}, \bullet p_1)$ by $(\mathcal{A}_{\mathtt{main}}, I)$. We will describe a winning strategy for Eve. We denote the positions in this game (p_x, I_y); until one of the players reaches a significant transition, Adam's token remains in states in the big blue box, and Eve's token in states in $B_{\mathtt{top}}$. The strategy will maintain the following invariant:

If Adam's token is in some state $\blacksquare p_i$, then Eve's token is in I.

Actually, every possible strategy satisfies this invariant: Adam's token can only enter a state $\blacksquare p_i$ upon reading a letter in $E + F + y$. For all these letters, the transitions of Eve's token are of the form $B_{\mathtt{top}} \to I$ (or a significant transition).

The only positions where Eve has a choice to make is when her token is in state I and Adam gives letter a. We define Eve's choice in positions of the form $(\blacksquare p_i, I)$ over a. If $i \in E$, Eve stays in I, that is, she follows $I \xrightarrow{a} I$. If $i \in F$, Eve moves to the right, that is, she follows $I \xrightarrow{a} I_a$. Over all other positions of the game, we define Eve's choice arbitrary (note that in most of them she has no choice at all).

We show that this is a winning strategy. Assume that in a play, Adam reaches a significant transition. We show that Eve does too. Consider the last moment during the play when Adam's token is in a state of the form $\blacksquare p_i$, and let u_i be the suffix of the word of the play from this moment. We have that $u_i \in L_i$, and moreover, $u_i \in A^* i$ (as $\blacksquare p_i$ is not visited again). Assume that $i \in E$ (Eve decides to stay on the left part of $B_{\mathtt{top}}$). If u_i does not contain letter a, then it is of the form $u_i = c^+ i$ (this is immediate from the transitions in Figure 10a and in the i^{th} box in the bottom row of Figure 10b). Therefore, Eve sees the significant transition $I_c \xRightarrow{i}$ at the same time as Adam. If u_i contains letter a, then it must be of the form $u_i = u_0 a u_i' c i$, with $u_i' \in (b + c)^*$. Then, Adam necessarily follows the path:

$$\blacksquare p_i \xrightarrow{u_0} \cdot \xrightarrow{a} \blacksquare q_i \xrightarrow{u_i'} \cdot \xrightarrow{c} \blacksquare s_i \xRightarrow{i} .$$

On the left part of $B_{\mathtt{top}}$, Eve's token follows the path:

$$I \xrightarrow{u_0} \cdot \xrightarrow{a} I \xrightarrow{u_i'} \cdot \xrightarrow{c} I_c \xRightarrow{i},$$

as wanted.

Finally, assume that $i \in F$ (Eve decides to go to the right part of $B_{\mathtt{top}}$). In this case, u_i must be of the form $(b + c)^* a A^* b i$. Then Eve's token follows the path

$$I \xrightarrow{(b+c)^*} \cdot \xrightarrow{a} I_a \xrightarrow{A^*} \cdot \xrightarrow{b} I_b \xRightarrow{i},$$

and also produces a significant transition. ◀

We conclude that $\mathcal{A}_{\mathtt{main}}$ is simplified, finishing the proof of Theorem 12.

C Proofs in Section 5: Succinctness of $\mathcal{A}_{\mathtt{main}}$

▶ **Lemma 16.** *The automata $\mathcal{C}_{\mathtt{main}}$ is history deterministic, and recognises the complement of the language recognised by $\mathcal{A}_{\mathtt{main}}$.*

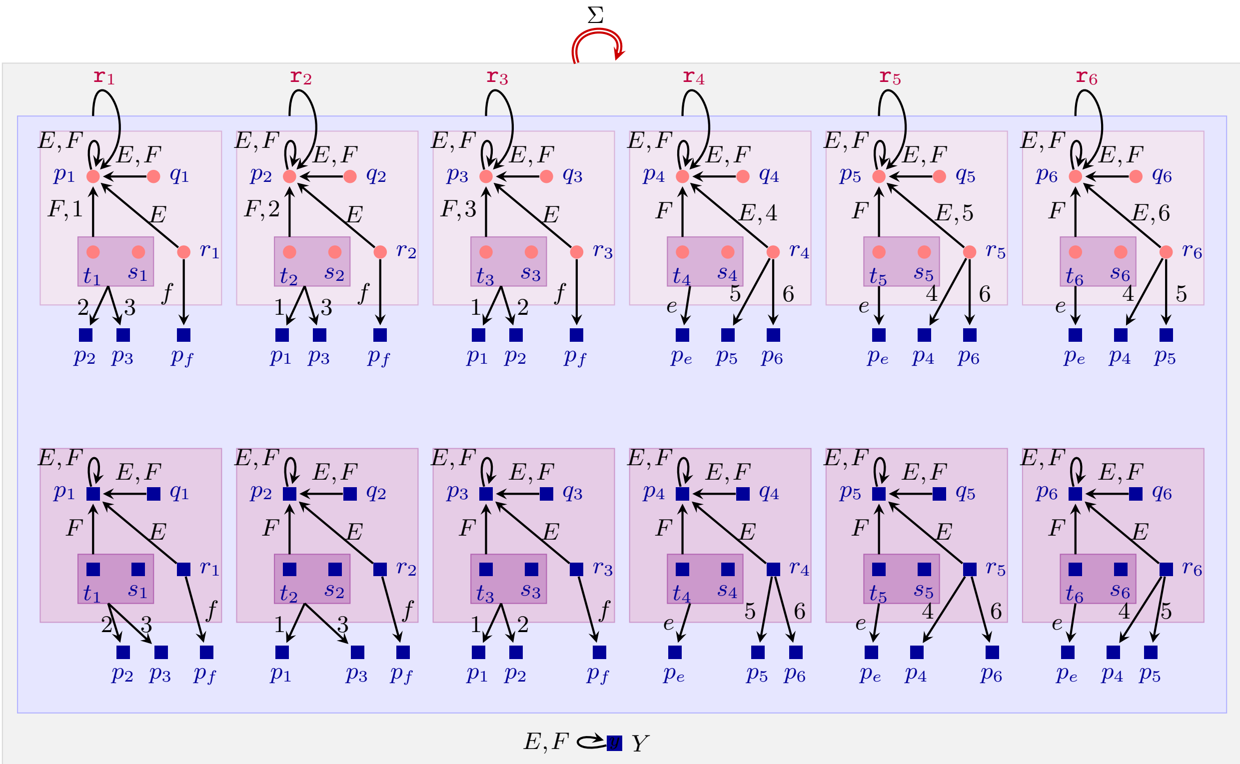


Figure 13 The HD coBüchi automaton $\mathcal{C}_{\mathtt{main}}$ obtained from the reach-deterministic states of $\mathcal{A}_{\mathtt{main}}$. Non-significant transitions are as the non-significant transitions in Figure 10. We omit transitions over a, b, c, y, which are exactly as in Figure 10a (note for instance that the states s_i are connected to the others by these transitions). As in Figure 10, we let $E = \{1, 2, 3\}$, $F = \{4, 5, 6\}$ and assume that e ranges from the set E and f from the set F. Note that over every state and every letter, there is a significant transition going to all states of the automaton (self-loop Σ on top).

Proof. We will prove that

1. $L(\mathcal{C}_{\mathtt{main}}) \subseteq L(\mathcal{A}_{\mathtt{main}})^c$,
2. $L(\mathcal{A}_{\mathtt{main}})^c \subseteq L(\mathcal{C}_{\mathtt{main}})$,
3. and that there is an HD-resolver for $\mathcal{C}_{\mathtt{main}}$.

Proof of 1. Let w be a word in $L(\mathcal{C}_{\mathtt{main}})$, and ρ be an accepting run of $\mathcal{C}_{\mathtt{main}}$ on w. Then, there is a suffix u of w and a state p in $\mathcal{C}_{\mathtt{main}}$ such that the (deterministic) run from p on u stays in the safe-component containing p and does not contain a significant transition. Then u is in $L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p)$, and thus, $u \notin L(\mathtt{reach}(\mathcal{A}_{\mathtt{main}}), p)$. Since $L(B) \subseteq L(\mathtt{reach}(B))$ for every Büchi automaton B, it follows that $u \notin L(\mathcal{A}_{\mathtt{main}}, p)$. Since $L(\mathcal{A}_{\mathtt{main}}, p)$ is prefix-independent and $\mathcal{A}_{\mathtt{main}}$ is semantically deterministic, $w \notin L(\mathcal{A}_{\mathtt{main}})$, as desired.

Proof of 2. Let $w = a_0 a_1 a_2 \dots$ be a word that is not in $L(\mathcal{A}_{\mathtt{main}})$. We claim that there is a state p of $\mathcal{A}_{\mathtt{main}}$ and a suffix $w_i = a_i a_{i+1} \dots$ of w such that every run of $(\mathcal{A}_{\mathtt{main}}, p)$ on w_i contains no significant transition. If this is not the case, then it is easy to inductively build a run of $\mathcal{A}_{\mathtt{main}}$ on w that contains infinitely many significant transitions, which contradicts the fact that $w \notin L(\mathcal{A}_{\mathtt{main}}, p)$.

Thus, $w_i \notin L(\mathtt{reach}(\mathcal{A}_{\mathtt{main}}), p)$ for some state p. We know, due to reach-covering for $\mathcal{A}_{\mathtt{main}}$, that there is a reach-deterministic state q such that p simulates q in $\mathtt{reach}(\mathcal{A}_{\mathtt{main}})$.

Then,

$$L(\texttt{reach}(\mathcal{A}_{\texttt{main}}), p) \supseteq L(\texttt{reach}(\mathcal{A}_{\texttt{main}}), q),$$

and thus $w_i \notin L(\texttt{reach}(\mathcal{A}_{\texttt{main}}), q)$. This implies, by construction of $\mathcal{C}_{\texttt{main}}$, that w_i is in $L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, q)$. Thus, $w \in L(\mathcal{C}_{\texttt{main}})$, since there is a run ρ of $\mathcal{C}_{\texttt{main}}$ on w that takes arbitrary transitions on $a_0 a_1 \dots a_{i-1}$, takes the (possibly significant) transition to q on a_i, and then stays in the safe-component containing q by taking deterministic non-significant transitions.

Proof of 3. The fact that $\mathcal{C}_{\texttt{main}}$ is HD follows from the fact that it is safe-deterministic and has significant transitions going everywhere. We explicitly describe a resolver γ for $\mathcal{C}_{\texttt{main}}$. The resolver γ, after reading the prefix $u = a_0 a_1 \dots a_{i-1}$, will be in some state q_i. On the letter a_i, γ takes the deterministic non-significant transition $q_i \xrightarrow{a_i} q_{i+1}$, if it exists. Otherwise, γ takes the significant transition $q_i \overset{a_i}{\Longrightarrow} p$ to a state p in $\mathcal{C}_{\texttt{main}}$, where p is the state with the longest suffix v of ua_i such that there is a run of $\texttt{safe}(\mathcal{C}_{\texttt{main}})$ on v that starts at some state p' and ends at p.

If the word w on which γ constructs a run is in $L(\mathcal{C}_{\texttt{main}})$, then there is a state p and a suffix w' of w for which $w' \in L(\texttt{safe}(\mathcal{C}_{\texttt{main}}), p)$. The run of γ on w can only contain finitely many significant transitions before it eventually coincides with the deterministic run of $(\mathcal{C}_{\texttt{main}}, p)$ on w' consisting of only non-significant transitions. Thus, γ is an HD-resolver for $\mathcal{C}_{\texttt{main}}$, as desired. ◀

▶ **Lemma 17.** *The automaton $\mathcal{C}_{\texttt{main}}$ is the statewise minimal HD coBüchi automaton recognising $L(\mathcal{C}_{\texttt{main}})$.*

Proof. To apply Lemma 14, we need to prove that $\mathcal{C}_{\texttt{main}}$ is HD, semantically deterministic, safe-deterministic, normal, safe-minimal, and safe-centralised. We argued that $\mathcal{C}_{\texttt{main}}$ is HD in Lemma 16, $\mathcal{C}_{\texttt{main}}$ is SD because every pair of states has a transition from one to the other on every letter (self-loop Σ), and $\mathcal{C}_{\texttt{main}}$ is safe-deterministic and normal by construction. We thus need to argue that $\mathcal{C}_{\texttt{main}}$ is safe-centralised and safe-minimised.

In the following, $L_{\texttt{safe}}(p)$ denotes $L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, p)$, for a state p in $\mathcal{C}_{\texttt{main}}$.

Safe-centralised. We note that there are only two safe-components in $\mathcal{C}_{\texttt{main}}$: the safe-component consisting of the state $\{Y\}$, and the safe-component consisting of the other 60 states, which we call $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$. Note that $L_{\texttt{safe}}(Y) = (\Sigma \setminus \{y\})^\omega$, while $y^\omega \in L_{\texttt{safe}}(s)$ for every state s in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$. Thus,

$$L_{\texttt{safe}}(s) \nsubseteq L_{\texttt{safe}}(Y)$$

for every state $s \in \mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$.

Similarly, we note that for each state s in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$, there is a word $w \in (\Sigma \setminus \{y\})^*$ such that $w \notin L_{\texttt{safe}}(s)$, but w is in $L_{\texttt{safe}}(y)$ (as it does not contain y). It follows that

$$L_{\texttt{safe}}(Y) \nsubseteq L_{\texttt{safe}}(s)$$

for every state $s \in \mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$. Thus, $\mathcal{C}_{\texttt{main}}$ is safe-centralised.

Safe-minimised. We need to show that for every pair of distinct states $p', q' \in \mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$, $L_{\texttt{safe}}(p') \neq L_{\texttt{safe}}(q')$. We split the proof in the following 5 cases.

Case 1. $p' = \bullet\pi_i$ and $q' = \blacksquare\pi'_j$. Then, a a finite word v in L_j satisfies

$$yvy^\omega \in L(\texttt{safe}(\mathcal{C}_{\texttt{main}}, \bullet\pi_i)) \setminus L(\texttt{safe}(\mathcal{C}_{\texttt{main}}, \blacksquare\pi'_j)).$$

Case 2. $p' = \bullet\pi_i$ and $q' = \bullet\pi'_j$ with $i \neq j$. Then, a word $v_j \in L_j$ satisfies

$$yv_jv_jy^\omega \in L_{\texttt{safe}}(\bullet\pi_i) \setminus L_{\texttt{safe}}(\blacksquare\pi'_j).$$

Case 3. $p' = \blacksquare\pi_i$ and $q' = \blacksquare\pi'_j$ with $i \neq j$. Then, a word v_j in L_j satisfies

$$yv_jy^\omega \in L_{\texttt{safe}}(\bullet\pi_i) \setminus L_{\texttt{safe}}(\blacksquare\pi'_j).$$

Case 4. $p' = \bullet\pi_i, q' = \bullet\pi'_i$, for some $\pi \neq \pi' \in \{p, q, r, s, t\}$. Let $e \in E = \{1, 2, 3\}$, $f \in F = \{4, 5, 6\}$, be such that $e, f \neq i$. We have the following 5 subcases here.

Case 4a. $\pi \in \{p, t\}, \pi' \in \{q, r, s\}$. Then, the word bf is such that $\bullet\pi_i \xrightarrow{bf} \bullet p_i$, and $\bullet\pi'_i \xrightarrow{bf} \blacksquare p_f$. Since $L_{\texttt{safe}}(\bullet p_i) \neq L_{\texttt{safe}}(\blacksquare p_f)$ from Case 1, we obtain that $L_{\texttt{safe}}(\bullet\pi_i) \neq L_{\texttt{safe}}(\bullet\pi'_i)$.

Case 4b. $\pi = p, \pi' = t$. Then, $\bullet\pi_i \xrightarrow{e} \bullet p_i$, whereas $\bullet\pi'_i \xrightarrow{e} \blacksquare p_e$. We conclude by Case 1.

Case 4c. $\pi = q, \pi' = r$. Then $\bullet\pi_i \xrightarrow{f} \bullet p_i$, whereas $\bullet\pi'_i \xrightarrow{f} \blacksquare p_f$. We conclude by Case 1.

Case 4d. $\pi = q, \pi' = s$. Then, $\bullet\pi_i \xrightarrow{e} \bullet p_i$, whereas $\bullet\pi'_i \xrightarrow{e} \blacksquare p_e$. We conclude by Case 1.

Case 4e. $\pi = r, \pi' = s$. Then, $\bullet\pi_i \xrightarrow{f} \blacksquare p_f$, and $\bullet\pi'_i \xrightarrow{f} \bullet p_i$. We conclude by Case 1.

Case 5. $p' = \blacksquare\pi_i, q' = \blacksquare\pi'_i$, with $\pi \neq \pi' \in \{p, q, r, s, t\}$. Let $e \in E, f \in F$ be such that $e, f \neq i$. We have the following subcases here.

Case 5a. $\pi \in \{p, t\}, \pi' \in \{q, r, s\}$. Then, the word bf is such that $\blacksquare\pi_i \xrightarrow{bf} \blacksquare p_i$, and $\blacksquare\pi'_i \xrightarrow{bf} \blacksquare p_f$. Since $L_{\texttt{safe}}(\blacksquare p_i) \neq L_{\texttt{safe}}(\blacksquare p_f)$ from Case 3, we obtain that $L_{\texttt{safe}}(\blacksquare\pi_i) \neq L_{\texttt{safe}}(\blacksquare\pi'_i)$.

Case 5b. $\pi = p, \pi' = t$. Then, $\blacksquare\pi_i \xrightarrow{e} \blacksquare p_i$, whereas $\blacksquare\pi'_i \xrightarrow{e} \blacksquare p_e$. We conclude by Case 3.

Case 5c. $\pi = q, \pi' = r$. Then $\blacksquare\pi_i \xrightarrow{f} \blacksquare p_i$, whereas $\blacksquare\pi'_i \xrightarrow{f} \blacksquare p_f$. We conclude by Case 3.

Case 5d. $\pi = q, \pi' = s$. Then, $\blacksquare\pi_i \xrightarrow{f} \blacksquare p_i$, whereas $\blacksquare\pi'_i \xrightarrow{e} \blacksquare p_e$. We conclude by Case 3.

Case 5e. $\pi = r, \pi' = s$. Then, $\blacksquare\pi_i \xrightarrow{f} \blacksquare p_f$, and $\blacksquare\pi'_i \xrightarrow{f} \blacksquare p_i$. We conclude by Case 3.

Thus, $\mathcal{C}_{\texttt{main}}$ is safe-minimal, safe-centralised, semantically deterministic, HD, and normal, and thus statewise minimal Lemma 14. ◀

We finally prove that $\mathcal{D}_{\texttt{main}}$ requires at least 66 states.

▶ **Lemma 29.** *If $\mathcal{D}_{\texttt{main}}$ has at most 65 states, then it does not recognise $L(\mathcal{C}_{\texttt{main}})$.*

Proof. We let $\mathsf{Fin}(\mathcal{K})$ (resp. $\mathsf{Fin}(\mathcal{K}_y)$) to be the language of finite words that are prefixes of words in $\mathcal{K}$ (resp. $\mathcal{K}_y$). First, note the following:

▷ Claim 33. For every word $u \in \mathsf{Fin}(\mathcal{K})$, there is a word $u' \in \{a, b, c, 1, \ldots, 6\}^*$ such that $(uu')^{-1}\mathcal{K} = \mathcal{K}$. That is, u' resets the automaton in Figure 11 to the initial state.

We leverage Lemma 28 to force a run to enter $\mathcal{D}_{\mathsf{core}}$ while reading prefixes of $\mathcal{K}$.

▷ Claim 34. For every state q outside $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$, there is a word $u \in \mathsf{Fin}(\mathcal{K})$ such that the run $q \xrightarrow{u}$ ends in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$.

Proof. By Lemma 28, if q is outside $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$, there is a word $u_1 \in \mathsf{Fin}(\mathcal{K}) \setminus L_{\texttt{safe}}(\mathcal{D}_{\texttt{main}}, q)$. By the previous claim, we can moreover assume that $(u_1)^{-1}\mathcal{K} = \mathcal{K}$. Then, the run $q \xrightarrow{u_1} q_1$ crosses some significant transition. While q_i is not in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$, we can repeat this process. If we never enter $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$, this produces a word in $\mathcal{K}$ whose run over $\mathcal{D}_{\texttt{main}}$ is rejecting, a contradiction. ◁

▷ Claim 35. For every state q in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ outside $\mathcal{D}_{\mathsf{core}}$, there is a word $u \in \mathsf{Fin}(\mathcal{K})$ such that the run $q \xrightarrow{u}$ ends in $\mathcal{D}_{\mathsf{core}}$.

Proof. Let $q_0 = q$. We produce a run

$$q_0 \xrightarrow{u_0} q_1 \xrightarrow{u_1} q_2 \xrightarrow{u_2} \dots,$$

such that, while $q_i \notin \mathcal{D}_{\mathsf{core}}$, $u_i \in \mathsf{Fin}(\mathcal{K})$ such that $u_i^{-1}\mathcal{K} = \mathcal{K}$ and either:

1. The state q_i is in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ and $u_i \in \mathsf{Fin}(\mathcal{K}_y) \setminus L_{y\mathtt{safe}}(q_i)$. In this case, either $q_i \xrightarrow{u_i}$ changes of y-SCC or visits a significant transition.
2. The state q_i is not in $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ and $q_i \xrightarrow{u_i}$ visits a significant transition.

The first case can be ensured by the second part of Lemma 28 and Claim 33. The second case is given by the previous claim. Since $\mathcal{D}_{\mathsf{core}}$ is the only end-y-SCC of $\mathcal{S}^{\mathcal{D}}_{\mathsf{safe}}$ (Lemma 24), it holds that if we never enter $\mathcal{D}_{\mathsf{core}}$, this procedure produces an infinite rejecting run over a word in $\mathcal{K}$, a contradiction. ◁

Let $\{o_1, \dots, o_5\}$ be the (at most) 5 states outside $\mathcal{D}_{\mathsf{core}}$. Combining the two previous claims, we get that there are words $u_1, \dots, u_5 \in \mathsf{Fin}(\mathcal{K})$ such that $o_i \xrightarrow{u_i} q_i$ ends in $\mathcal{D}_{\mathsf{core}}$. Let p_i in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ such that $L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q_i) = L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p_i)$ (such a state exists by Lemma 22). We may assume (by Claim 33) that moreover each p_i is of the form $\bullet p_j$ or $\blacksquare p_j$ for some j. Let $Q_{\mathsf{target}} = \{q_1, \dots, q_5\}$ the set of possible destinations of the above paths and $P_{\mathsf{target}} = \{p_1, \dots, p_5\} \subseteq \{\bullet p_j, \blacksquare p_j \mid j \in [6]\}$ the set (of size at most 5) of corresponding safe-equivalent states in $\mathcal{C}_{\mathtt{main}}$. Let j_0 be an index such that $\bullet p_{j_0}, \blacksquare p_{j_0} \notin P_{\mathsf{target}}$.

▷ Claim 36. For every state q in Q_{target} there is a word $v \in L^2_{j_0}$ such that the run $q \xrightarrow{v}$ ends outside $\mathcal{D}_{\mathsf{core}}$.

Proof. Let p in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$ such that $L_{\mathtt{safe}}(\mathcal{D}_{\mathtt{main}}, q) = L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, p)$ and $p \in \{\bullet p_i, \blacksquare p_i\}$, with $i \neq j_0$. Take a word $v \in L^2_{j_0}$ that does not contain y. Then, the run $p \xrightarrow{v}$ in $\mathcal{C}_{\mathtt{main}}$ produces a significant transition (indeed, $p \xrightarrow{L_{j_0}} \blacksquare p_{j_0} \xRightarrow{L_{j_0}}$, see Figure 13). By equivalence of safe-languages, v produces a significant transition during the run $q \xRightarrow{v} q'$. Assume by contradiction that q' is in $\mathcal{D}_{\mathsf{core}}$. By definition, $\mathcal{D}_{\mathsf{core}}$ is a y-SCC, therefore, there is path $q' \xrightarrow{v'} q$ containing no y. We obtain that $(vv')^\omega$ is rejected by $\mathcal{D}_{\mathtt{main}}$, as we have found a cycle over vv' containing a significant transition. This contradicts the fact that $\mathcal{C}_{\mathtt{main}}$ only rejects words that contain y infinitely often. ◁

Using the above remarks we build a rejecting run of the form

$$o_{i_1} \xrightarrow{u_{i_1}} q_{i_1} \xRightarrow{v} o_{i_2} \xrightarrow{u_{i_2}} q_{i_2} \xRightarrow{v} \dots,$$

where $u_i \in \mathsf{Fin}(\mathcal{K})$ and $v \in L^2_{j_0}$. However, this word is accepted by $\mathcal{C}_{\mathtt{main}}$, because it does not contain a factor of the form $L_{\alpha\beta\beta}$, for $\alpha \neq \beta$. ◀

D Proof of Lemma 25: Computer-aided lemmas

In this section, we will prove the following result by the aid of computers, namely, the tool DFAMiner [DLS24].

▶ **Lemma 25.** *Let $\mathcal{D}$ be a deterministic safety automaton that separates $\mathcal{K}$ (resp. $\mathcal{K}_y$) and $L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, q)$ for some state q in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$, i.e.,*

$$\mathcal{K} \subseteq L(\mathcal{D}) \subseteq L_{\mathtt{safe}}(\mathcal{C}_{\mathtt{main}}, q).$$

Then, $\mathcal{D}$ has at least 5 states.

We note that for two safety automata $\mathcal{A}$ and $\mathcal{B}$, $L(\mathcal{A}) \subseteq L(\mathcal{B})$ if and only if $\mathsf{Fin}(L(\mathcal{A})) \subseteq \mathsf{Fin}(L(\mathcal{B}))$, where $\mathsf{Fin}(L)$ is the set of prefixes of words in L. Thus, in this section, we view our safety automata as DFAs, where every state is accepting, together with an additional rejecting sink state for completion.

Towards proving Lemma 25, we will first restrict candidates for our state q in $\mathcal{S}^{\mathcal{C}}_{\mathsf{safe}}$. Let d_0 be the initial state of $\mathcal{D}$. Let $q \xrightarrow{1} q'$, and $d_0 \xrightarrow{1} d'$. Note that $1 \equiv_L \varepsilon$ in $\mathcal{K}$ and $\mathcal{K}_y$. Thus, $(\mathcal{D}, d')$ separates $\mathcal{K}$ and $L_{\texttt{safe}}(\mathcal{C}_{\texttt{main}}, q')$. But note that q' is either $\bullet p_i$ or $\blacksquare p_i$, for some $i \in [6]$. Furthermore, in $\mathcal{A}_{\texttt{main}}$, the role of $\{1, 2, 3\}$ is symmetric, as well as $\{4, 5, 6\}$. Thus, it suffices to prove Lemma 25 for when q' is a state in $\{\bullet p_2, \bullet p_5, \blacksquare p_1, \blacksquare p_4\}$.

▶ Remark 37. Our choices for $2, 5, 1$, and 4 here might seem arbitrary, but it is essential for when we restrict the alphabet set to $\{a, b, c, 1, 4\}$ next. Lemma 39 is not true for $q = \bullet p_1$ or $q = \bullet p_4$.

For a DFA $\mathcal{A}$ over the alphabet Σ and a subset $\Gamma \subseteq \Sigma$, we use $\mathcal{A}^\Gamma$ to denote the DFA obtained by restricting the transitions of $\mathcal{A}$ to the alphabet Γ. We make another simplification, based on the following simple observation.

▶ **Proposition 38.** *Let $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ be three DFA over Σ such that*

$$L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq L(\mathcal{C}).$$

Then, for every $\Gamma \subseteq \Sigma$,

$$L(\mathcal{A}^\Gamma) \subseteq L(\mathcal{B}^\Gamma) \subseteq L(\mathcal{C}^\Gamma).$$

For a language L over infinite words, we use L_Γ to denote the set of words in $L \cap \Gamma^\omega$. Due to Proposition 38, it suffices to show the following result.

▶ **Lemma 39.** *Let $\Gamma = \{a, b, c, 1, 4\}$, and let $\mathcal{D}$ be a DFA that separates $\mathsf{Fin}(\mathcal{K}_\Gamma)$ and $\mathsf{Fin}(L_{\texttt{safe}}(\mathcal{C}^\Gamma_{\texttt{main}}, q))$, for some $q \in \{\bullet p_2, \bullet p_5, \blacksquare p_1, \blacksquare p_4.\}$ Then, $\mathcal{D}$ has at least 6 states (accounting for a rejecting sink state).*

Note that in Lemma 39, the size of $(\mathcal{C}^\Gamma_{\texttt{main}}, q)$ is 15 for each q, which is much smaller than 60. This allows us to use the tool DFAMiner.

D.1 Using DFAMiner

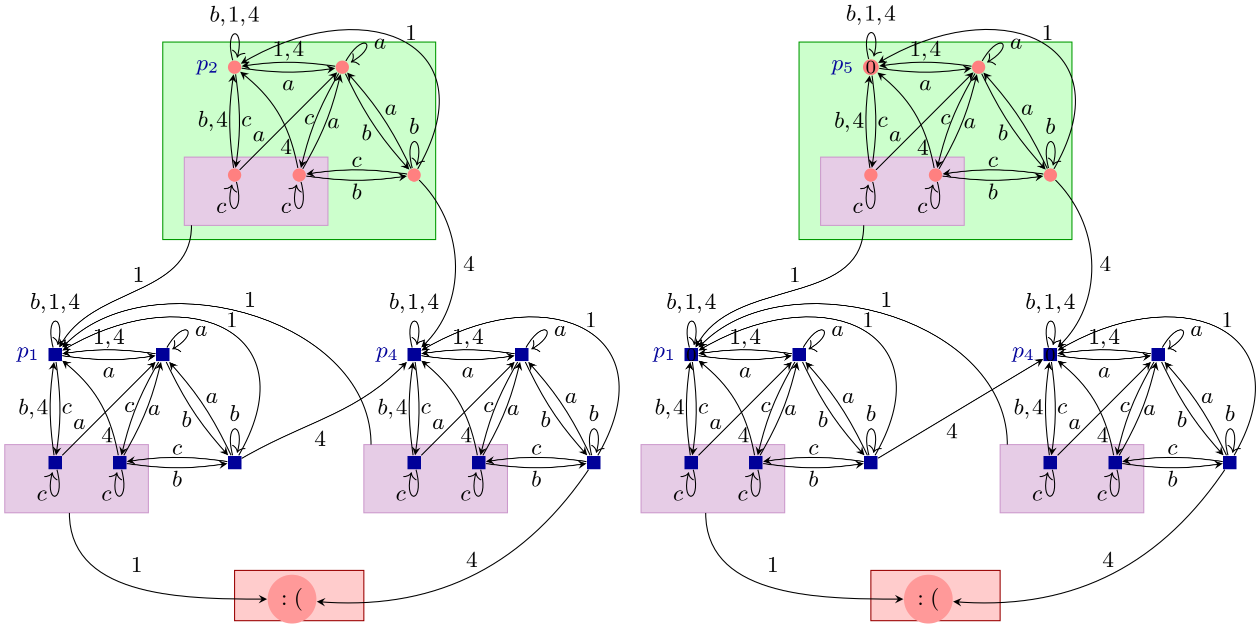
One of the features of DFAMiner is that it minimises 3-DFAs. A 3-DFA has the syntax of a DFA, but each state is labelled as either accept, reject, or do not care. The idea is that a 3-DFA $\mathcal{A}$ accepts a word w if the run of $\mathcal{A}$ on w ends at an accepting state and rejects w if the run of $\mathcal{A}$ on w ends at rejecting sink state.

The problem of 3-DFA minimisation is the following: Given a 3-DFA $\mathcal{A}$, construct the smallest DFA $\mathcal{B}$ such that $\mathcal{B}$ accepts every word accepted by $\mathcal{A}$ and $\mathcal{B}$ rejects every word rejected by $\mathcal{A}$.

It is easy to see that the problem of separating regular languages can be reduced to the problem of minimising a 3-DFA that is obtained by a product construction. For proving our Lemma 39, this product construction results in 16 state DFAs, due to the similarity of structure between the automaton recognising $\mathcal{K}$ and $\mathcal{C}_{\texttt{main}}$. The four 3-DFAs that we need to consider are depicted in Figures 14a, 14b, 15a, and 15b.

The tool DFAMiner, when asked to minimise these four 3-DFAs, outputs that the minimal 3-DFA has size 6, which coincides with the deterministic safety automaton recognising $\mathcal{K}_y$ together with a rejecting sink state. We attach an encoding of these automata in the desired format of the DFAMiner tool for the readers who wish to verify.

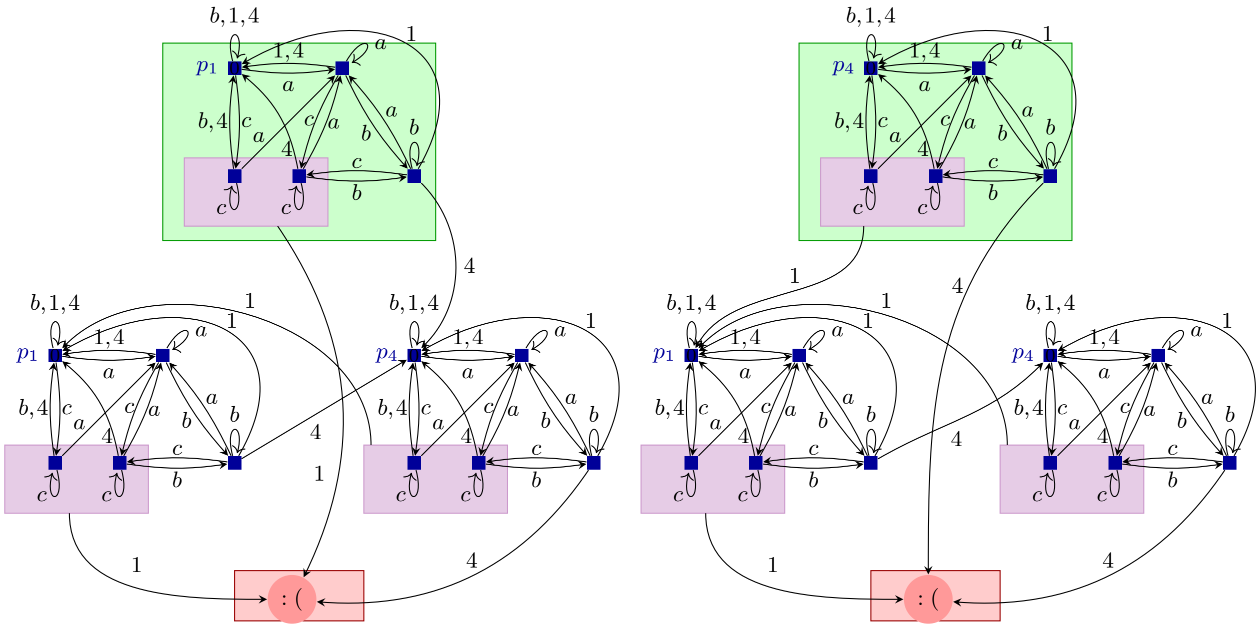
D.2 Figures of the four 3-DFAs



(a) The 3-DFA for when q is $\bullet p_2$.

(b) The 3-DFA for when q is $\bullet p_5$.

Figure 14 3-DFAs for states $\bullet p$. States in the green box are accepting; the red box is rejecting.



(a) The 3-DFA for when q is $\blacksquare p_1$.

(b) The 3-DFA for when q is $\blacksquare p_4$.

Figure 15 The 3-DFAs for states $\blacksquare p$.

D.3 Encoding of 3-DFAs for DFAMiner

D.3.1 3-DFA for when q is $\bullet p_2$

```

16 5 -- number of states,
-- alphabet: 0 a, 1 b,
-- 2 c, 3 1, 4 4

i 0 -- initial state

t 0 0 1 -- transistion from 0 on 0 to 1
t 0 1 0
t 0 2 2
t 0 3 0
t 0 4 0

t 1 0 1
t 1 1 4
t 1 2 3
t 1 3 0
t 1 4 0

t 2 0 1
t 2 1 0
t 2 2 2
t 2 4 0

t 3 0 1
t 3 1 4
t 3 2 3
t 3 4 0


t 4 0 1
t 4 1 4
t 4 2 3
t 4 3 0



t 5 0 6
t 5 1 5
t 5 2 7
t 5 3 5
t 5 4 5

t 6 0 6
t 6 1 9
t 6 2 8
t 6 3 5
t 6 4 5

t 7 0 6
t 7 1 5
t 7 2 7
t 7 4 5

t 8 0 6
t 8 1 9
t 8 2 8
t 8 4 5

t 9 0 6
t 9 1 9
t 9 2 8
t 9 3 5

t 10 0 11
t 10 1 10
t 10 2 12
t 10 3 10
t 10 4 10

t 11 0 11
t 11 1 14
t 11 2 13
t 11 3 10
t 11 4 10

t 12 0 11
t 12 1 10
t 12 2 12
t 12 4 10

t 13 0 11
t 13 1 14
t 13 2 13
t 13 4 10


t 14 0 11
t 14 1 14
t 14 2 13
t 14 3 10



t 2 3 5
t 3 3 5
t 4 4 10

t 7 3 15
t 8 3 15
t 9 4 10

t 12 3 5
t 13 3 5
t 14 4 15

--accepting states
a 0
a 1
a 2
a 3
a 4
--rejecting states
r 15
```

D.3.2 3DFA for when q is $\bullet p_5$

```
16 5 -- number of states,
-- alphabet: 0 is a, 1 is b,
-- 2 is c, 3 is 1, 4 is 4

i 0 -- initial state

t 0 0 1 -- transistion from 0 on 0 to 1
t 0 1 0
t 0 2 2
t 0 3 0
t 0 4 0

t 1 0 1
t 1 1 4
t 1 2 3
t 1 3 0
t 1 4 0

t 2 0 1
t 2 1 0
t 2 2 2
t 2 4 0

t 3 0 1
t 3 1 4
t 3 2 3
t 3 4 0


t 4 0 1
t 4 1 4
t 4 2 3
t 4 3 0



t 5 0 6
t 5 1 5
t 5 2 7
t 5 3 5
t 5 4 5

t 6 0 6
t 6 1 9
t 6 2 8
t 6 3 5
t 6 4 5

t 7 0 6
t 7 1 5
t 7 2 7
t 7 4 5

t 8 0 6
t 8 1 9
t 8 2 8
t 8 4 5

t 9 0 6
t 9 1 9
t 9 2 8
t 9 3 5

t 10 0 11
t 10 1 10
t 10 2 12
t 10 3 10
t 10 4 10

t 11 0 11
t 11 1 14
t 11 2 13
t 11 3 10
t 11 4 10

t 12 0 11
t 12 1 10
t 12 2 12
t 12 4 10

t 13 0 11
t 13 1 14
t 13 2 13
t 13 4 10


t 14 0 11
t 14 1 14
t 14 2 13
t 14 3 10



t 2 3 5
t 3 3 5
t 4 4 10

t 7 3 15
t 8 3 15
t 9 4 10

t 12 3 5
t 13 3 5
t 14 4 15

--accepting states
a 0
a 1
a 2
a 3
a 4
--rejecting states
r 15
```

D.3.3 3-DFA for when q is $\blacksquare p_1$

```
    16 5 -- number of states,
-- alphabet: 0 is a, 1 is b,
-- 2 is c, 3 is 1, 4 is 4

i 0 -- initial state

t 0 0 1 -- transistion from 0 on 0 to 1
t 0 1 0
t 0 2 2
t 0 3 0
t 0 4 0

t 1 0 1
t 1 1 4
t 1 2 3
t 1 3 0
t 1 4 0

t 2 0 1
t 2 1 0
t 2 2 2
t 2 4 0

t 3 0 1
t 3 1 4
t 3 2 3
t 3 4 0


t 4 0 1
t 4 1 4
t 4 2 3
t 4 3 0



t 5 0 6
t 5 1 5
t 5 2 7
t 5 3 5
t 5 4 5

t 6 0 6
t 6 1 9
t 6 2 8
t 6 3 5
t 6 4 5

t 7 0 6
t 7 1 5
t 7 2 7
t 7 4 5

t 8 0 6
t 8 1 9
t 8 2 8
t 8 4 5

t 9 0 6
t 9 1 9
t 9 2 8
t 9 3 5

t 10 0 11
t 10 1 10
t 10 2 12
t 10 3 10
t 10 4 10

t 11 0 11
t 11 1 14
t 11 2 13
t 11 3 10
t 11 4 10

t 12 0 11
t 12 1 10
t 12 2 12
t 12 4 10

t 13 0 11
t 13 1 14
t 13 2 13
t 13 4 10


t 14 0 11
t 14 1 14
t 14 2 13
t 14 3 10



t 2 3 15
t 3 3 15
t 4 4 10

t 7 3 15
t 8 3 15
t 9 4 10

t 12 3 5
t 13 3 5
t 14 4 15

--accepting states
a 0
a 1
a 2
a 3
a 4
--rejecting states
r 15
```

D.3.4 3-DFA for when q is $\blacksquare p_4$

```
16 5 -- number of states,
-- alphabet: 0 is a, 1 is b,
-- 2 is c, 3 is 1, 4 is 4

i 0 -- initial state

t 0 0 1 -- transistion from 0 on 0 to 1
t 0 1 0
t 0 2 2
t 0 3 0
t 0 4 0

t 1 0 1
t 1 1 4
t 1 2 3
t 1 3 0
t 1 4 0

t 2 0 1
t 2 1 0
t 2 2 2
t 2 4 0

t 3 0 1
t 3 1 4
t 3 2 3
t 3 4 0


t 4 0 1
t 4 1 4
t 4 2 3
t 4 3 0



t 5 0 6
t 5 1 5
t 5 2 7
t 5 3 5
t 5 4 5

t 6 0 6
t 6 1 9
t 6 2 8
t 6 3 5
t 6 4 5

t 7 0 6
t 7 1 5
t 7 2 7
t 7 4 5

t 8 0 6
t 8 1 9
t 8 2 8
t 8 4 5

t 9 0 6
t 9 1 9
t 9 2 8
t 9 3 5

t 10 0 11
t 10 1 10
t 10 2 12
t 10 3 10
t 10 4 10

t 11 0 11
t 11 1 14
t 11 2 13
t 11 3 10
t 11 4 10

t 12 0 11
t 12 1 10
t 12 2 12
t 12 4 10

t 13 0 11
t 13 1 14
t 13 2 13
t 13 4 10


t 14 0 11
t 14 1 14
t 14 2 13
t 14 3 10



t 2 3 5
t 3 3 5
t 4 4 15

t 7 3 15
t 8 3 15
t 9 4 10

t 12 3 5
t 13 3 5
t 14 4 15

--accepting states
a 0
a 1
a 2
a 3
a 4
--rejecting states
r 15
```