UNIVERSITY OF UDINE

DOCTORAL THESIS
XXXVI CYCLE

Automata and orders: the Wheeler class. Complexity, bounds, and operations

Author:
Davide MARTINCIGH

Supervisor:
Alberto POLICRITI
Co-supervisor:
Giovanna D'AGOSTINO

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in

Computer Science, Mathematics and Physics
DMIF

September 30, 2024

"*People love to talk about the early bird, but nobody ever mentions the fate of the early worm.*"

Shel Silverstein, *Where the Sidewalk Ends*

UNIVERSITY OF UDINE

Abstract

Computer Science, Mathematics and Physics
DMIF

Doctor of Philosophy

Automata and orders: the Wheeler class. Complexity, bounds, and operations

by Davide MARTINCIGH

In this thesis we discuss Wheeler automata, a subclass of finite states automata (NFAs) ordered by co-lexicographic order on strings, which allows efficient storage and substring query mechanisms. Wheeler automata form an important data structure for languages, as the determinization process via powerset construction is polynomial, making classical problems solvable in polynomial time.

We investigate computational problems related to recognizing Wheeler automata starting from NFAs and *reduced* NFAs, noting that non-determinism generally leads to intractability. We also examine state complexity in Wheeler DFAs and prove that intersection of languages is computationally simpler, and we provide a construction for the minimum Wheeler DFA.

Additionally, we explore the Krohn-Rhodes Decomposition Theorem (KRDT) for two compression-oriented classes of automata: path-coherent and Wheeler automata. These classes are efficiently encodable and indexable. We prove that automata in these classes can be decomposed into a cascade with a number of components linear to the original automaton's states. For Wheeler automata, only two-state resets are needed, avoiding full KRDT through a simpler inductive argument.

Lastly, we extend the analysis of Wheeler automata to arbitrary NFAs using a parameter called *width*, which indicates how far an automaton is from being Wheeler. Specifically, we focus on the difference between the width calculated on DFAs and that calculated on NFAs recognizing a given language, showing that their distance can be exponentially large and providing useful lower bounds for the latter.

Acknowledgements

I could not have undertaken this journey without my supervisors, Alberto Policriti and Giovanna D'Agostino, whom I thank for their constant support throughout my PhD. Thank you for encouraging and pushing me through the dark moments of my journey, and for the everlasting patience that I particularly appreciated as various deadlines approached. I feel I have received more support than I deserved, and for this, I am especially grateful.

Thank you to my family, who supported me in many ways and always believed in me. Thank you for always being there when I knew I needed you but also when I thought I didn't, for always listening to me and giving advice.

Thanks to my colleague Andrea Gulli, who had the honor of putting up with me and the Lobby for these long years. This journey would have been less fun and less interesting without you.

Finally, I would like to thank everyone who contributed, directly or indirectly, to the creation of this thesis, whom I have not mentioned here.

Contents

Introduction

This thesis combines two cardinal concepts of computer science, automata and orders, with the intent of analyzing what happens when we enhance the former with the latter. The first ingredient is *automata*. Finite state automata (NFAs) represent abstract models of computation of fundamental importance in computer science, since they are capable of capturing the essential dynamics of systems that exhibit discrete and sequential behavior. One of the primary appeals of NFAs lies in their elegance and accessibility. With a finite number of states and transitions between them, these automata provide a concise and intuitive framework for describing complex processes. This simplicity facilitates analysis, enabling us to reason rigorously about the behavior of systems and develop clever solutions. NFAs find utility in diverse areas such as artificial intelligence, bioinformatics, and cryptography, where modeling sequential processes and recognizing patterns are paramount.

The second ingredient is *orders*. In computer science, ordered data structures play a critical role in algorithm design and optimization. Ordered arrays, trees, and graphs form the basis for efficient data storage, retrieval, and manipulation, powering algorithms for search, sorting, and graph traversal. In this work we explore what happens if we impose a (reasonable) order to the states of a NFA. In the literature we find many different ways to impose a (partial) order on the states of a NFA. In [1, 2, 3] partially-ordered NFAs are defined as automata where the reachability relation induces a partial order on their states. In [4], instead, a total order preserved by pairs of transitions with the same label was introduced. In [5] a further requirement was introduced: the order of the states must follow the (fixed) order of the alphabet. This is the definition of *Wheeler graphs*, a simple and unified perspective on several algorithmic techniques related to *suffix sorting* (in particular, to the Burrows-Wheeler transform [6], an ubiquitous string permutation in text compression and indexing — see also [7]). Wheeler graphs admit efficient data structures for solving string matching on the graph's paths and enable a representation of the graph in a space proportional to that of the edges' labels — as well as enabling more advanced compression mechanisms, see [8, 9, 10, 11, 12, 13, 14, 15].

The definition of Wheeler graphs has been lifted to automata in [16], where it has been proved that we can decide in polynomial time whether a deterministic automaton (DFA) is Wheeler. In contrast, in [17] it was proved that deciding whether a NFA is Wheeler is a NP-complete problem. In [16] it has been analyzed how this specific notion of order on automata influences the class of recognized languages, the Wheeler languages. This class of languages exhibits several positive properties that reflect those of regular languages. For instance, the class of languages recognized by Wheeler NFAs (WNFAs) coincides with the one recognized by Wheeler DFAs (WDFAs). Moreover, given a Wheeler language there exists a unique (up to isomorphism) WDFA that recognizes the language while minimizing the number of states: as a matter of fact, it is possible to prove a convex version of the Myhill-Nerode theorem suited for Wheeler languages. Some properties are actually an improvement on the one of regular languages: it is well-known that, despite recognizing the same class of languages, a DFA recognizing a regular language can

be exponentially larger than a NFA recognizing the same language. This is not the case for Wheeler automata where the distance between a WDFA and a WNFA recognizing the same language can always be made linear, avoiding the blow-up of the number of states. This result is of significant importance because it allows, in various decision problems, to easily overcome the difficulties intrinsically linked to non-determinism, often allowing the design of algorithms that work in polynomial time. In [18], the complexity of the powerset construction algorithm on families of subregular languages is categorized in great depth. This study proves that the output of the powerset construction is exponential in the size of the input NFA for the most widely studied classes of subregular languages, such as ordered [4], star-free [19], comet [20], and suffix/prefix/infix-closed languages. In the worst case, the resulting DFA for each of the aforementioned classes may have at least $2^n - 1$ states, where n is the number of states of the input NFA. The class of finite languages [21] and unary regular languages [22] are two examples of previously recognized families with a sub-exponential upper bound. Nonetheless, the former is a subclass of Wheeler languages and the latter is very restrictive regarding the types of languages it contains.

Another important result for this class is that the membership problem is decidable, and in polynomial time if the language is presented through a DFA. This result is nontrivial because, if a language is Wheeler, it is not guaranteed that its minimum automaton is Wheeler. However, there exists a combinatorial property of the minimum automaton that allows us to decide whether the language it recognizes is Wheeler. In [16] it was still open the problem of determining the complexity of deciding whether a NFA recognizes a Wheeler language, which we will address in this thesis.

One of the weaknesses of Wheeler automata derives from the fact that their definition requires the existence of a total order on the set of states. As one might expect, a consequence of this restrictive requirement is that not all automata are Wheeler, and indeed, it has been shown that Wheeler automata are a subclass of the relatively small class of counter-free automata [16]. In an attempt to transfer the excellent properties of Wheeler automata to any NFA, a parameter called the width of an automaton, denoted by p, was introduced in [23]. This parameter, in a sense, measures how far an automaton is from being Wheeler. More precisely, if we allow a partial order to be defined on the set of states of an NFA $\mathcal{A}$, the width of $\mathcal{A}$ measures how far the "best" order on the states of $\mathcal{A}$ is from being total (where by "best" we mean the one that comes closest to being total). In [23], it was shown that several problems solvable in polynomial time on Wheeler automata and in non-polynomial time on general NFAs turn out to be fixed parameter tractable (FPT) using p as a parameter; that is, they can be solved in polynomial time under the assumption that p is a constant. Some examples, among others, are the following: given an NFA $\mathcal{A}$ with n states and width p,

- the well-known explosion in the number of states occurring when computing the powerset DFA $\mathrm{Pow}(\mathcal{A})$ is exponential in p, rather than in n;
- $\mathcal{A}$ can be encoded using just $\Theta(|\Sigma| \log p)$ bits per edge, rather than $\Theta(|\Sigma| \log n)$;
- string matching on $\mathcal{A}$'s paths and testing membership in the automaton's accepted language can be solved in time proportional to p^2 per matched character.

In this thesis we will address various problems related to Wheelerness, divided in three macro topics.

1. Complexity. In Chapter 2, we study the complexity of various problems related to Wheeler automata and languages. In particular, we examine the state complexity of some operations between WDFAs, showing that better bounds are obtained compared to DFAs. Additionally, we describe an algorithm that allows us to transition from the minimum DFA recognizing a language $\mathcal{L}$ to the minimum WDFA recognizing it. This transition is non-trivial because, in general, the size of the minimum WDFA can be exponentially larger than the one of the minimum DFA. Finally, we study the complexity of some classic problems (e.g. membership and universality) for two special classes of NFAs.

   The first class is that of Generalized Wheeler automata (GWNFAs), where any possible ordering of the starting alphabet is allowed. The definition of WNFAs, in fact, assumes that the starting alphabet has a prefixed order, and it is natural to ask the following questions: can changing the order of the alphabet change the Wheelerness of an automaton? If so, how complex is the problem of deciding whether there exists an appropriate ordering of the alphabet that makes an NFA Wheeler? How do these results reflect on the class of languages recognized by GWNFAs?

   The second class is that of *reduced* NFAs, i.e., automata without "redundant" states, where by "redundant" states we mean pairs of states reached by the same set of strings. These pairs of states are redundant because they can be collapsed together into a single super-state to obtain an NFA with one less state that recognizes the same language. This class of NFAs is interesting because in [16] it was shown that Wheeler membership for reduced NFAs is decidable in polynomial time, unlike that for general NFAs. In this thesis, we address an open question, namely whether reduced NFAs allow us to decide other Wheeler-related problems in polynomial time.

2. Krohn-Rhodes Decomposition Theorem (KRDT). In Chapter 3 we focus on KRDT [24], a central result in automata and semigroup theories: it states that any (deterministic) finite-state automaton can be disassembled into a collection of automata of two simple types, that can be arranged into a combination – *cascade* – that simulates the original automaton. The elementary building blocks of the decomposition are either *resets* or *permutations*.

   The full-fledged theorem features two orthogonal dimensions of complexity: the type and the number of building blocks appearing in the cascade, and a deep step in the proof is the characterization of the permutations appearing in the decomposition. This characterization implies, in the case of *counter-free* automata, that the resulting cascade contains no permutations [25, 26].

   We start analysing KRDT for two compression-oriented classes of automata: (i) *path coherent*: state-ordered automata mapping state-intervals to state-intervals; (ii) Wheeler. In [5] Wheeler automata were proved to be a subclass of path coherent automata.

   We prove that each automata in these classes can be decomposed as a cascade with a number of components which is *linear* in the number of states of the original automaton and, for the Wheeler class, we prove that only two-state resets are needed. Our line of reasoning avoids the necessity of using full KRDT and proves our results directly by a simple inductive argument.

3. Width. In Chapter 4, we focus on non-Wheeler NFAs, specifically those with width $p \geq 2$. We study two natural extensions of the concept of width to

languages, defined as follows: the deterministic (non-deterministic) width of a language $\mathcal{L}$ is the smallest among the widths of the DFAs (NFAs) that recognize $\mathcal{L}$. It has been proved in [23] that these two measures do not always coincide: there are languages for which the non-deterministic width is strictly smaller than the deterministic one. Moreover, it has also been shown in [23] that for every integer $n \geq 1$, there exists a language whose deterministic width coincides with its non-deterministic width, and both are equal to n. An open problem, which is addressed in this thesis, concerns the possible distance between these two measures: we know they can differ, but by how much? We will prove that the distance between these two measures can be made arbitrarily large[1].

Additionally, we will analyze the notion of *entanglement* for DFAs as given in [23]. This is a measure closely related to deterministic width: it has been shown that the entanglement of the minimum automaton of a language $\mathcal{L}$ coincides with the deterministic width of $\mathcal{L}$. In this thesis, we will extend the definition of entanglement by attempting to define a similar measure that, when calculated on the minimum automaton, captures the non-deterministic width of a language. Unfortunately, this task is challenging because the reference minimum automaton is deterministic, whereas we are trying to capture a measure calculated on non-deterministic automata. Using the minimum DFA seems to be unavoidable for two reasons: the first one is the lack of uniqueness (up to isomorphism) of the minimum NFA that recognizes a regular language. The second one is that, in general, the NFA that achieves the minimum width is not one of those that minimize the number of states; often, to reduce the width, it is necessary to duplicate states to "untangle" the strings that reach them. Nonetheless, we prove that these new measures (both computed on the minimum DFA) are lower bounds to the non-deterministic width of a language.

All the results explicitly proven in this thesis are original work. With the exception of the one contained in Chapter 4, they have all been published in the following articles:

- Giovanna D'Agostino, Davide Martincigh, and Alberto Policriti. "Ordering Regular Languages: a Danger Zone". Proceedings of the 22nd Italian Conference on Theoretical Computer Science, Bologna, Italy, September 13-15, 2021. Vol. 3072, pag. 46-69.
  https://ceur-ws.org/Vol-3072/paper5.pdf.
- Giovanna D'Agostino, Davide Martincigh, and Alberto Policriti. "Ordering regular languages and automata: Complexity". Theoretical Computer Science, 2023. Vol. 949.
  https://doi.org/10.1016/j.tcs.2023.113709.
- Giovanna D'Agostino, Luca Geatti, Davide Martincigh, and Alberto Policriti. "A Linear-size Cascade Decomposition for Wheeler Automata". Proceedings of the 24th Italian Conference on Theoretical Computer Science, Palermo, Italy, September 13-15, 2023. Vol. 3587, pag. 181-191.
  https://ceur-ws.org/Vol-3587/0763.pdf.
- Giovanna D'Agostino, Luca Geatti, Davide Martincigh, and Alberto Policriti. "Cascade Products and Wheeler Automata". Theoretical Computer

[1] Without exceeding the upper bound shown in [23], where it was proved that the deterministic width is at most exponentially larger than the non-deterministic one.

Science, 2024. Vol. 1013.
https://doi.org/10.1016/j.tcs.2024.114754.

Chapter 1

Notation and literature results

1.1 Automata and orders

We assume that the reader has a good understanding of automata theory, as described for example in [27]. Nevertheless, we will present the necessary results and notation.

We denote by $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$ a finite automaton (NFA), with Q as set of states, q_0 initial state, $\delta : Q \times \Sigma \to 2^Q$ transition function, and $F \subseteq Q$ final states. Sometimes we shall describe the transition function δ using edges, where the edge (q, q', a) stands for $q' \in \delta(q, a)$. The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is defined to be $|Q|$. An automaton is deterministic (DFA) if $|\delta(q, a)| \leq 1$[1], for all $q \in Q$ and $a \in \Sigma$. As customary, we extend δ to operate on strings as follows: for all $q \in Q$, $a \in \Sigma$ and $\alpha \in \Sigma^*$

$$\delta(q, \varepsilon) = \{q\}, \qquad \delta(q, \alpha a) = \bigcup_{v \in \delta(q,\alpha)} \delta(v, a).$$

We assume that every automaton is *trimmed*, that is, every state is reachable from the initial state and every state can reach at least one final state. Note that this assumption is not restrictive, since removing every state not reachable from q_0 and every state from which is impossible to reach a final state from an NFA, can be done in linear time and does not change the accepted language. It immediately follows that:

- there might be only one state without incoming edges, namely q_0;
- every string that can be read starting from q_0 belongs to the set of prefixes, $\mathrm{Pref}(\mathcal{L})$, of the language $\mathcal{L}$.

We denote by $\mathcal{L}(\mathcal{A}) = \{\alpha \in \Sigma^* : \delta(q_0, \alpha) \cap F \neq \emptyset\}$ the language accepted (or recognized) by the automaton $\mathcal{A}$. We say that two automata are *equivalent* if they accept the same language. The languages accepted by automata form the class of *regular languages* and are closed under boolean operations (union, intersection, complementation), concatenation, and the Kleene star. Given two languages $\mathcal{L}_1, \mathcal{L}_2$ recognized by two DFAs we can build the intersection automaton and the union automaton recognizing the languages $\mathcal{L}_1 \cap \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2$ respectively. Both of these constructions rely on the direct product between the two automata, formally defined below.

Definition 1. Let $\mathcal{D}_1 = (Q^{\mathcal{D}_1}, q_0^{\mathcal{D}_1}, \delta^{\mathcal{D}_1}, F^{\mathcal{D}_1}, \Sigma)$, $\mathcal{D}_2 = (Q^{\mathcal{D}_2}, q_0^{\mathcal{D}_2}, \delta^{\mathcal{D}_2}, F^{\mathcal{D}_2}, \Sigma)$ be two DFAs recognizing the languages $\mathcal{L}_1, \mathcal{L}_2$ respectively. The *direct product* of $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \times \mathcal{D}_2$, is the triple $(Q^{\mathcal{D}_1 \times \mathcal{D}_2}, \delta^{\mathcal{D}_1 \times \mathcal{D}_2}, \Sigma)$ defined as follows.

[1] Notice that, for us, deterministic DFAs do not need to be complete, i.e. the transition function can be partial.

- $Q^{\mathcal{D}_1 \times \mathcal{D}_2} := Q^{\mathcal{D}_1} \times Q^{\mathcal{D}_2}$ is the set of states;
- $\delta^{\mathcal{D}_1 \times \mathcal{D}_2} : Q^{\mathcal{D}_1 \times \mathcal{D}_2} \to Q^{\mathcal{D}_1 \times \mathcal{D}_2}$ is the function defined, for each $(q_1, q_2) \in Q^{\mathcal{D}_1 \times \mathcal{D}_2}$, as
$$\delta^{\mathcal{D}_1 \times \mathcal{D}_2}(q_1, q_2) := \big(\delta^{\mathcal{D}_1}(q_1), \delta^{\mathcal{D}_2}(q_2)\big).$$

The *intersection automaton* is the DFA

$$\big(Q^{\mathcal{D}_1 \times \mathcal{D}_2}, (q_0^{\mathcal{D}_1}, q_0^{\mathcal{D}_2}), \delta^{\mathcal{D}_1 \times \mathcal{D}_2}, F^{\mathcal{D}_1} \times F^{\mathcal{D}_2}, \Sigma\big),$$

which recognises the intersection language $\mathcal{L}_1 \cap \mathcal{L}_2$.
The *union automaton* is the DFA

$$\big(Q^{\mathcal{D}_1 \times \mathcal{D}_2}, (q_0^{\mathcal{D}_1}, q_0^{\mathcal{D}_2}), \delta^{\mathcal{D}_1 \times \mathcal{D}_2}, F^{\mathcal{D}_1} \times Q^{\mathcal{D}_2} \cup Q^{\mathcal{D}_1} \times F^{\mathcal{D}_2}, \Sigma\big),$$

which recognises the union language $\mathcal{L}_1 \cup \mathcal{L}_2$.

We will often make use of the notion of the *input* and *output language* of a state of an NFA, defined as follows.

Definition 2 (Input and output language)**.** Let $\mathcal{A} = (Q, q_0, \delta, F)$ be a NFA and let q be a state in Q. We say that the *input language* of q, denoted by I_q, is the set of strings leaving q_0 and entering q. Formally, I_q is the language recognized by the automaton $(Q, q_0, \delta, \{q\})$.
Similarly, we say that the *output language* of q, denoted by O_q, is the set of strings leaving q and entering a final state. Formally, O_q is the language recognized by the automaton (Q, q, δ, F).

As it is well known, every NFA is equivalent to a DFA obtained by the *powerset construction*.

Definition 3 (Powerset construction)**.** Let $\mathcal{A} = (Q, q_0, \delta, F)$ be a NFA. The *determinization* of $\mathcal{A}$, obtained used a technique called *powerset construction*, is the trimmed version of the DFA $\mathcal{D} = (Q^{\mathcal{D}}, q_0^{\mathcal{D}}, \delta^{\mathcal{D}}, F^{\mathcal{D}})$ defined as follows.

- $Q^{\mathcal{D}} := \mathrm{Pow}(Q)$. That is, the set of states of $\mathcal{D}$ is the set of all possible subsets of Q.
- $q_0^{\mathcal{D}} := \{q_0\}$.
- For each $S \in \mathrm{Pow}(Q)$ and for each character c in the alphabet, the transition function is defined as
$$\delta^{\mathcal{D}}(S, c) := \bigcup_{q \in S} \delta(q, c).$$
- $F^{\mathcal{D}} := \{S \in \mathrm{Pow}(Q) \mid S \cap F \neq \emptyset\}$.

Notice that some of the states of the DFA $\mathcal{D}$ in Definition 3 might not be reachable, hence the requirement of trimming it.

One of the main basic results of automata theory is the fact that, given a regular language, there exists a unique up to isomorphism, state-wise minimum DFA that recognizes such language. The states of this minimum DFA correspond to the classes of the following equivalence relation.

Definition 4 (Myhill-Nerode equivalence)**.** Let $\mathcal{L} \subseteq \Sigma^*$ be a language. Given a string $\alpha \in \Sigma^*$, we define the *right context* of α as

$$\alpha^{-1}\mathcal{L} := \{\gamma \in \Sigma^* : \alpha\gamma \in \mathcal{L}\},$$

and we denote by $\equiv_{\mathcal{L}}$ the Myhill-Nerode equivalence on $\mathrm{Pref}(\mathcal{L})$ defined as

$$\alpha \equiv_{\mathcal{L}} \beta \iff \alpha^{-1}\mathcal{L} = \beta^{-1}\mathcal{L}.$$

The following theorem guarantees the uniqueness of the minimum automaton.

Theorem 1 (Myhill-Nerode)**.** *Given a language $\mathcal{L} \subseteq \Sigma^*$, the following are equivalent:*

1. *$\mathcal{L}$ is a regular language (i.e. $\mathcal{L}$ is recognized by a NFA).*
2. *$\equiv_{\mathcal{L}}$ has finite index.*
3. *$\mathcal{L}$ is a union of classes of a right invariant equivalence over Σ^* of finite index.*
4. *$\mathcal{L}$ is recognized by a DFA.*

If $\mathcal{L}$ is a regular language we will denote by $\mathcal{D}_{\mathcal{L}}$ its minimum automaton, which can be defined using Theorem 1 as shown in the following proposition.

Proposition 2. *Given a regular language $\mathcal{L} \subseteq \Sigma^*$, there exist a unique (up to isomorphism) state-wise minimum automaton $\mathcal{D}_{\mathcal{L}} = (Q, q_0, \delta, F)$ defined as follows.*

- *$Q := \{[\alpha]_{\mathcal{L}} \mid \alpha \in \mathrm{Pref}(\mathcal{L})\}$, where $[\alpha]_{\mathcal{L}}$ is the Myhill-Nerode equivalence class of the string α (see Definition 4).*
- *$q_0 := [\varepsilon]_{\mathcal{L}}$.*
- *For each string α and for each character a in the alphabet, the transition function is defined as*
$$\delta([\alpha]_{\mathcal{L}}, a) := [\alpha \cdot a]_{\mathcal{L}}.$$
- *$F := \{[\alpha]^c_{\mathcal{L}} \mid \alpha \in \mathcal{L}\}$.*

Notice that the automaton $\mathcal{D}_{\mathcal{L}}$ in Proposition 2 has a finite number of states due to condition 2 of Theorem 1. Moreover, the output languages (see Definition 2) of the states of $\mathcal{D}_{\mathcal{L}}$ must be pairwise distinct. In fact, assume that two states have the same output language: we can obtain a new automaton by erasing one of them and redirecting all its incoming edges to the second one. It is easy to prove that this new automaton recognizes the same language and clearly has one less state. Notice also that an immediate consequence of Proposition 2 is the fact that the set of input languages of the states of $\mathcal{D}_{\mathcal{L}}$ coincides with the Myhill-Nerode equivalence classes contained in $\mathcal{L}$.

In this thesis we will be mainly concerned with a specific class of automata, the Wheeler auotomata, and the class of languages they recognize, Wheeler languages. As we shall see, these classes are subclasses of the following very well-known and studied classes of NFAs and regular languages.

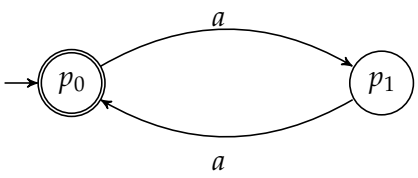
Definition 5 (Counter-free automata and star-free languages)**.** Let $\mathcal{A} = (Q, q_0, \delta, F)$ be a DFA. A sequence of states $p_0, p_1, \ldots, p_k$ with $k \geq 1$ is called a *counter* if and only if $p_i \neq p_j$ for all $0 \leq i < j \leq k$ and there exists a string α such that $\delta(p_i, \alpha) = p_{i+1}$ for all $0 \leq i \leq k$, where $p_{k+1} := p_0$.

A DFA is called *counter-free* if and only if it has no counters. A regular language is called *star-free* if can be obtained starting from finite languages and using all boolean operations and concatenation.

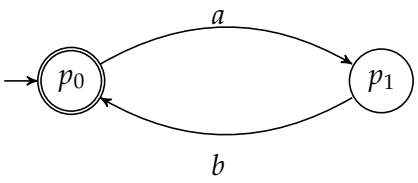
In [28] it was proved that the language recognized by a counter-free DFA is star-free. Moreover, to decide whether a regular language is star-free one can use the following theorem, proved again in [28]. We will see that a similar result holds for the class of Wheeler languages in Theorem 16.

Theorem 3. *A regular language is star-free if and only if its minimum DFA is counter-free.*

In Figure 1.1 are depicted two minimum automata: one is counter-free, hence recognizing a star-free language, whereas the other has counters, hence recognizing a language that is not star-free.



(a) The minimum DFA $\mathcal{D}_{\mathcal{L}_1}$ recognizing the language $\mathcal{L}_1 = (aa)^*$, containing the counter p_0, p_1 (with $\alpha = a$).



(b) The minimum DFA $\mathcal{D}_{\mathcal{L}_2}$ recognizing the language $\mathcal{L}_2 = (ab)^*$. It is easy to check that $\mathcal{D}_{\mathcal{L}_2}$ contains no counters.

FIGURE 1.1: Since $\mathcal{D}_{\mathcal{L}_1}$ contains a counter, the language $\mathcal{L}_1$ is not star-free; clearly, $\mathcal{L}_1$ is able to "count" modulo 2. Conversely, $\mathcal{D}_{\mathcal{L}_2}$ does not contain a counter hence the language $\mathcal{L}_2$ is star-free. As a matter of fact, we can write $\mathcal{L}_2 = (ab)^*$ as $(a\Sigma^* \cap \Sigma^* b) \setminus (\Sigma^* aa\Sigma^* \cup \Sigma^* bb\Sigma^*)$; if we replace Σ^* with $\bar{\emptyset}$ (that is, the complement of the empty set) in the previous expression we obtain a regular expression for $\mathcal{L}_2$ that makes no use of the Kleene star.

1.2 Wheeler automata and languages

In this section we will state some results related to Wheeler automata, a subclass of NFAs whose states admit a special order. Wheeler automata are the focus of this entire thesis but, before giving the formal definition, we need to fix some notation on orders.

We assume that there is a fixed total order $\preceq$ on the alphabet Σ of finite automata (in our examples, the alphabetical order). We extend $\preceq$ to strings in Σ^* *co-lexicographically*, that is, for $\alpha, \beta \in \Sigma^*$, we have $\alpha \preceq \beta$ if and only if either α is a suffix of β, or there exist $\alpha', \beta', \gamma \in \Sigma^*$ and $a, b \in \Sigma$, such that $\alpha = \alpha' a \gamma$ and $\beta = \beta' b \gamma$ and $a \prec b$. In the following, we will always compare strings of characters using the co-lexicographic order instead of the lexicographic one. This is due to the fact that the order of the states of a Wheeler automaton, which will be defined later, induces an order on the strings reaching them, but only if we compare the strings co-lexicographically.

Given two strings $\alpha, \beta \in \Sigma^*$, we denote by $\alpha \dashv \beta$ the property that α is a suffix of β.

If $(Z, \leq)$ is a partial order, we denote by $(Z, <)$ its corresponding strict partial order. Given a partial order $(Z, \leq)$ we say that a subset $I \subseteq Z$ is *convex* if, for any $x, y, z \in Z$ with $x < y < z$, if $x, z \in I$ then $y \in I$.

The class of Wheeler automata has been recently introduced in [29]. An automaton in this class has the property that there exists a total order on its states that is propagated along equally labeled transition. Moreover, the order must be compatible with the underlying order of the alphabet:

Definition 6 (Wheeler Automaton)**.** A Wheeler NFA (WNFA) $\mathcal{A}$ is a NFA $(Q, q_0, \delta, F, \Sigma)$ endowed with a binary relation $<$ such that: $(Q, <)$ is a linear order having the initial state q_0 as minimum, q_0 has no in-going edges, and the following two (Wheeler) properties are satisfied. Let $v_1 \in \delta(u_1, a_1)$ and $v_2 \in \delta(u_2, a_2)$:

(W1) $a_1 \prec a_2 \rightarrow v_1 < v_2$

(W2) $(a_1 = a_2 \wedge u_1 < u_2) \rightarrow v_1 \leq v_2$.

A Wheeler DFA (WDFA) is a WNFA in which the cardinality of $\delta(u, a)$ is always less than or equal to one.

In Figure 1.2 is depicted an example of a WDFA.

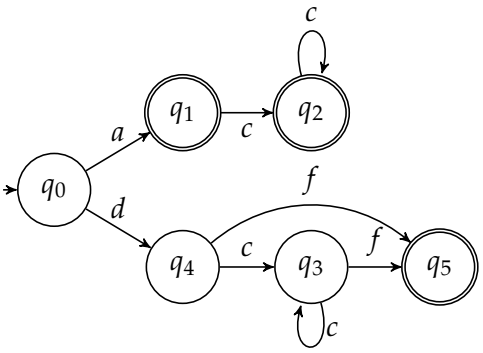


FIGURE 1.2: A WDFA $\mathcal{D}$ recognizing the language $\mathcal{L}_d = ac^* \cup dc^*f$. Condition W1 of Definition 6 implies input consistency and induces the partial order $q_1 < q_2, q_3 < q_4 < q_5$. From condition W2 it follows that $\delta(q_1, c) \leq \delta(q_4, c)$, thus $q_2 < q_3$. Therefore, the only order that could make $\mathcal{D}$ Wheeler is $q_0 < q_1 < q_2 < q_3 < q_4 < q_5$. The reader can verify that condition W2 holds for each pair of equally labeled edges.

Remark 4. A consequence of Wheeler property (W1) is that $\mathcal{A}$ is *input-consistent*, that is all transitions entering a given state $u \in Q$ have the same label: if $u \in \delta(v, a)$ and $u \in \delta(w, b)$, then $a = b$. Note that, for a fixed (i.e. constant in size) alphabet, requiring an automaton to be *input-consistent* is not computationally demanding. In fact, given an NFA $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$ we can build an equivalent, input-consistent one just by creating, for each state $q \in Q$, at most $|\Sigma|$ copies of q, that is, one for each different incoming label of q. This operation can be performed in $O(|Q| \cdot |\Sigma|)$ time.

A nice property of WNFAs is that the Wheeler order of its states is reflected on the "order" of their input languages. To define such an order over languages, which are sets of strings, we give the following general definition.

Definition 7 (Prefix-suffix relation)**.** Let $(L, \leq)$ be a total order and let I_1, I_2 be two subsets of L. We define the *prefix-suffix* relation among subsets of L as follows:

$$I_1 \leq_{ps} I_2 \iff \forall \alpha \in I_1 \forall \beta \in I_2 \big(\{\alpha, \beta\} \not\subseteq I_1 \cap I_2 \rightarrow \alpha \prec \beta\big).$$

Proposition 5 ([23]). *Let $\mathcal{A}$ be a WNFA with set of states Q and Wheeler order $<$. Then, given two states $q, q' \in Q$,*

$$q \leq q' \implies I_q \preceq_{ps} I_{q'},$$

where $\preceq_{ps}$ is the prefix-suffix order induced by the co-lexicographic order $(\Sigma^*, \prec)$.

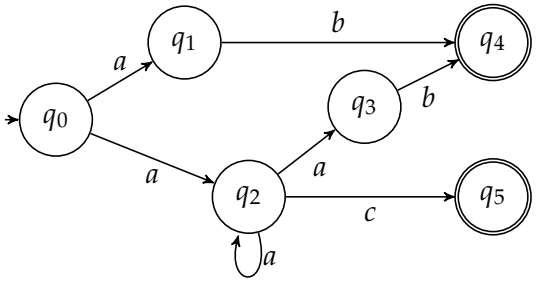


FIGURE 1.3: A WNFA $\mathcal{A}$. The Wheeler order of its states satisfying (W1) and (W2) is $q_0 < q_1 < q_2 < q_3 < q_4 < q_5$.

Example 1. Consider the WNFA $\mathcal{A}$ in Figure 1.3. The input languages of the states of $\mathcal{A}$ are

$$I_{q_0} = \{\varepsilon\},\ I_{q_1} = \{a\},\ I_{q_2} = a \cdot a^*,\ I_{q_3} = aa \cdot a^*,\ I_{q_4} = a \cdot a^* b,\ I_{q_5} = a \cdot a^* c.$$

Just by looking at the common ending character of the strings in each input language we can immediately verify that the implication $q \leq q' \implies I_q \preceq_{ps} I_{q'}$ holds for most pairs of sates. For instance, all strings in I_{q_3} end with a and all strings in I_{q_5} end with c, hence $I_{q_3} \prec_{ps} I_{q_5}$. The only non-trivial inequalities to check are $I_{q_1} \preceq_{ps} I_{q_2} \preceq_{ps} I_{q_3}$. By applying Definition 7 we have $I_{q_1} \preceq_{ps} I_{q_2}$ iff

$$\forall \alpha \in I_{q_1} \forall \beta \in I_{q_2} \big(\{\alpha, \beta\} \not\subseteq I_{q_1} \cap I_{q_2} \to \alpha \prec \beta \big).$$

Since a is the only string in I_{q_1} the formula above is equivalent to

$$\forall \beta \in I_{q_2} \setminus \{a\}\ (a \prec \beta),$$

which is indeed true. Similarly, by applying Definition 7 we have $I_{q_2} \preceq_{ps} I_{q_3}$ iff

$$\forall \alpha \in I_{q_2} \forall \beta \in I_{q_3} \big(\{\alpha, \beta\} \not\subseteq I_{q_2} \cap I_{q_3} \to \alpha \prec \beta \big).$$

Since a is the only string in I_{q_2} not belonging to $I_{q_2} \cap I_{q_3}$ the formula above is equivalent to

$$\forall \beta \in I_{q_3} \setminus \{a\}\ (a \prec \beta),$$

which is true. Therefore, the input languages are ordered as follows:

$$I_{q_0} \prec_{ps} I_{q_1} \prec_{ps} I_{q_2} \prec_{ps} I_{q_3} \prec_{ps} I_{q_4} \prec_{ps} I_{q_5}.$$

Consider the previous result restricted to DFAs. Since given two states $q \neq q'$ of a DFA it always holds $I_q \cap I_{q'} = \emptyset$, the condition $I_q \prec_{ps} I_{q'}$ translates to $\forall \alpha \in I_q \forall \beta \in I_{q'} (\alpha \prec \beta)$. Therefore we have the following proposition, where we extend the definition of $\prec$ to set of strings in the natural way: for $X, Y \subseteq \Sigma^*$, we denote by

$X \prec Y$ the relation defined as

$$\forall \alpha \in X \forall \beta \in Y (\alpha \prec \beta).$$

Proposition 6 ([23]). *Let $\mathcal{D}$ be a DFA. Then the relation*

$$q <_{\mathcal{D}} q' \iff I_q \prec I_{q'}$$

is a partial order. Moreover, this order is total if and only if $\mathcal{D}$ is Wheeler.

This result is important because it makes possible to decide in polynomial time whether a DFA is Wheeler: for each state q, pick the shortest string α_q entering it and order the states reflecting the co-lexicographic order of the strings $\{\alpha_q : q \in Q\}$; then check if the order satisfies the Wheeler conditions.

Lemma 7 ([30]). *The Wheeler Automata Recognition problem is in P for DFAs.*

In contrast, in [17] the following Theorem was proved.

Theorem 8. *The Wheeler Automata Recognition problem is NP-complete for NFAs.*

The proof of this Theorem consists of a reduction from the Betweenness problem, which is stated below and has been shown to be NP-complete [31].

Definition 8 (Betweenness). Input: a list Y of n distinct elements $Y = y_1, ..., y_n$ and $k < n^3$ ordered triples $(a_1, b_1, c_1), ..., (a_k, b_k, c_k)$ each composed of three different elements belonging to Y.
Output: yes/no answer. The answer is "yes" if and only if there exists a total order $<$ of Y such that, for each k, either $a_k < b_k < c_k$ or $a_k > b_k > c_k$.

Among the many NP-complete problems studied in the literature, Betweenness is not one of the most popular: it involves ordering elements, which is rarely a crucial characteristic of combinatoric problems. It should not be surprising that this problem is related to Wheelerness, where we are interested in ordering the states of an automaton. In fact, we will also use the Betweenness problem in Section 2.3 to prove the NP-completeness of a new problem that emerges when we drop the restriction that the alphabet's order is fixed.

Another interesting theorem about Wheeler automata is the following, which states that the exponential blow-up in the number of states associated to the power-set construction does not occur on WNFAs.

Theorem 9. *(see [16]) If $\mathcal{A} = (Q, s, \delta, <, F)$ is a WNFA with $|Q| = n$ and $\mathcal{L} = \mathcal{L}(\mathcal{A})$, then there exists a unique minimum-size WDFA $\mathcal{B}$ with at most $2n - 1 - |\Sigma|$ states such that $\mathcal{L} = \mathcal{L}(\mathcal{B})$.*

Corollary 9.1. *The class of languages recognized by WNFAs coincides with the class of languages recognized by WDFAs.*

In [29] it is shown that WNFAs have a property called *path coherence*, that we shall use in Chapter 3.

Lemma 10 (Path coherence). *Let $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$ be a WNFA according to the order $(Q, <)$. Then for every interval of states $I = [q_i, q_j]$ and for all $\alpha \in \Sigma^*$, the set J of states reachable starting from any state in I by reading α is also an interval. In particular, given a string $\alpha \in \Sigma^*$ the set $I_\alpha = \delta(q_0, \alpha)$ is an interval. Moreover, given a state $q \in Q$ its input language I_q is a convex set in $(Pref(\mathcal{L}), \prec)$*

We now present an analogous of the classical Myhill-Nerode Theorem (1) for Wheeler languages. In order to state it, we replace the equivalence $\equiv_{\mathcal{L}}$ by the equivalence $\equiv^c_{\mathcal{L}}$ defined below.

Definition 9. The input-consistent, convex refinement $\equiv^c_{\mathcal{L}}$ of $\equiv_{\mathcal{L}}$ is defined as follows. $\alpha \equiv^c_{\mathcal{L}} \beta$ if and only if

- $\alpha \equiv_{\mathcal{L}} \beta$,
- α and β end with the same character,
- for all $\gamma \in \text{Pref}(\mathcal{L})$, if $\min(\alpha, \beta) \preceq \gamma \preceq \max(\alpha, \beta)$, then $\alpha \equiv_{\mathcal{L}} \gamma \equiv_{\mathcal{L}} \beta$.

Remark 11. Notice that $\equiv^c_{\mathcal{L}}$-classes are convex for any regular language $\mathcal{L}$, but in non-Wheeler languages there is an infinite number of them, as stated in the following theorem.

Theorem 12 (Myhill-Nerode for Wheeler Languages [30])**.** *Given a language $\mathcal{L} \subseteq \Sigma^*$, the following are equivalent:*

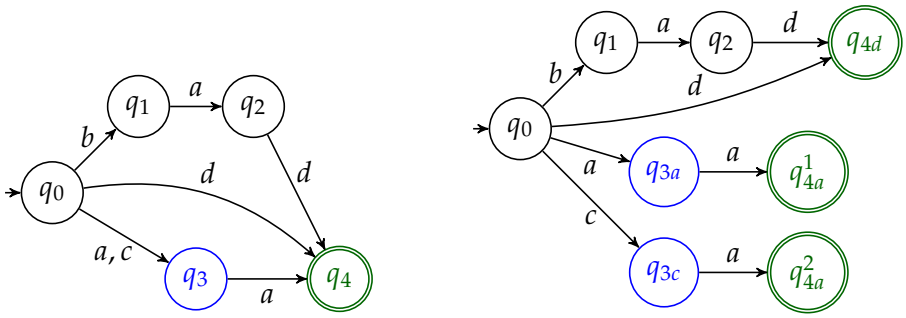
1. *$\mathcal{L}$ is a Wheeler language (i.e. $\mathcal{L}$ is recognized by a WNFA).*
2. *$\equiv^c_{\mathcal{L}}$ has finite index over Pref($\mathcal{L}$).*
3. *$\mathcal{L}$ is a union of classes of a convex, input-consistent, right invariant equivalence over Pref($\mathcal{L}$) of finite index.*
4. *$\mathcal{L}$ is recognized by a WDFA.*

Using the Myhill-Nerode Theorem for Wheeler languages we can prove that there exists a minimum (in the number of states) WDFA recognizing $\mathcal{L}$, and such WDFA is unique up to isomorphism with a construction similar to the one given in Proposition 2.

Proposition 13. *Given a regular language $\mathcal{L} \subseteq \Sigma^*$, there exist a unique (up to isomorphism) state-wise minimum WDFA $\mathcal{D}^W_{\mathcal{L}} = (Q, q_0, \delta, F)$ defined as follows.*

- *$Q := \{[\alpha]_{\equiv^c_{\mathcal{L}}} \mid \alpha \in Pref(\mathcal{L})\}$, where $[\alpha]_{\equiv^c_{\mathcal{L}}}$ denotes the equivalence class of the string α for the relation defined in Definition 9).*
- *$q_0 := [\varepsilon]_{\equiv^c_{\mathcal{L}}}$*
- *For each string α and for each character a in the alphabet, the transition function is defined as*
$$\delta\left([\alpha]_{\equiv^c_{\mathcal{L}}}, a\right) := [\alpha \cdot a]_{\equiv^c_{\mathcal{L}}}.$$
- *$F := \{[\alpha]_{\equiv^c_{\mathcal{L}}} \mid \alpha \in \mathcal{L}\}$.*

Remark 14. As in the classic case, the input languages of the states of the minimum WDFA are, in fact, the $\equiv^c_{\mathcal{L}}$-equivalence classes, this time consisting of convex sets of strings, and the Wheeler order between them is given by the $\prec$-order over subsets of strings. In particular, given a Wheeler language $\mathcal{L}$ and two distinct Wheeler classes W_1, W_2 of $\mathcal{L}$ (corresponding to the input languages of two distinct states of $\mathcal{D}^W_{\mathcal{L}}$) it always holds either $W_1 \prec W_2$ or $W_2 \prec W_1$.



(a) The minimum DFA $\mathcal{D}_\mathcal{L}$ recognizing $\mathcal{L}$.

(b) The minimum WDFA $\mathcal{D}_\mathcal{L}^W$ recognizing $\mathcal{L}$.

FIGURE 1.4: The minimum DFA and WDFA recognizing a Wheeler language $\mathcal{L}$. To obtain $\mathcal{D}_\mathcal{L}^W$, states q_3 and q_4 have been split due to input-consistency and convexity requirements.

In Figure 1.4, the minimum DFA and the minimum WDFA recognizing a Wheeler language $\mathcal{L}$ are depicted. Consider the Myhill-Nerode equivalence classes of $\mathcal{L}$, that is, the input languages of the states of $\mathcal{D}_\mathcal{L}$:

$$I_{q_0} = \{\varepsilon\}, \quad I_{q_1} = \{b\}, \quad I_{q_2} = \{ba\}, \quad I_{q_3} = \{a, c\}, \quad I_{q_4} = \{aa, ca, d, bad\}.$$

Starting from these classes, we can easily obtain the $\equiv_\mathcal{L}^c$-classes of Definition 9. First, we refine the equivalence in order to make these classes input-consistent, that is, each new class will contain strings ending with the same character. Therefore we split the class I_{q_3} into two classes, $I_{q_{3a}} = \{a\}$ and $I_{q_{3c}} = \{c\}$. We do the same for the class I_{q_4}, obtaining $I_{q_{4a}} = \{aa, ca\}$ and $I_{q_{4d}} = \{d, bad\}$. Second, we refine this input-consistent equivalence in order to make these classes convex. The only class that is not already convex is $I_{q_{4a}} = \{aa, ca\}$: as a matter of fact, we have $aa \prec ba \prec ca$ with $ba \in I_{q_2} \neq I_{q_{4a}}$. Therefore we split $I_{q_{4a}}$ into $I_{q_{4a}^1} = \{aa\}$ and $I_{q_{4a}^1} = \{ca\}$. We end up with 8 distinct input-consistent, convex, right invariant equivalence classes which will be the states of $\mathcal{D}_\mathcal{L}^W$. The edges are trivially inherited from $\mathcal{D}_\mathcal{L}$. Notice that in the previous example things were easy due to the fact that the language $\mathcal{L}$ considered is finite. We will see how to generalize this procedure to the general case using Algorithm 1 in Section 2.2.

An important consequence of the Myhill-Nerode Theorem for Wheeler languages, especially for testing Wheelerness, is stated in the following Lemma (proved in [16]).

Lemma 15. *A regular language $\mathcal{L}$ is Wheeler if and only if all monotone sequences in $(Pref(\mathcal{L}), \prec)$ become eventually constant modulo $\equiv_\mathcal{L}$. In other words, for all sequences $(\alpha_i)_{i\geq 0}$ in $Pref(\mathcal{L})$ with*

$$\alpha_1 \preceq \alpha_2 \preceq \ldots \alpha_i \preceq \ldots \quad \text{or} \quad \alpha_1 \succeq \alpha_2 \succeq \cdots \succeq \alpha_i \succeq \ldots$$

there exists an n such that $\alpha_h \equiv_\mathcal{L} \alpha_k$, for all $h, k \geq n$.

Lemma 15 shows how it is possible to recognize whether a language $\mathcal{L}$ is Wheeler simply by verifying a property on the strings of Pref($\mathcal{L}$): trying to find a WDFA that recognizes $\mathcal{L}$ is no longer needed to decide the Wheelerness of $\mathcal{L}$. As shown in Theorem 16 (see [16]), we can verify whether the property mentioned in Lemma 15 is satisfied just analysing the structure of the minimum DFA recognizing $\mathcal{L}$.

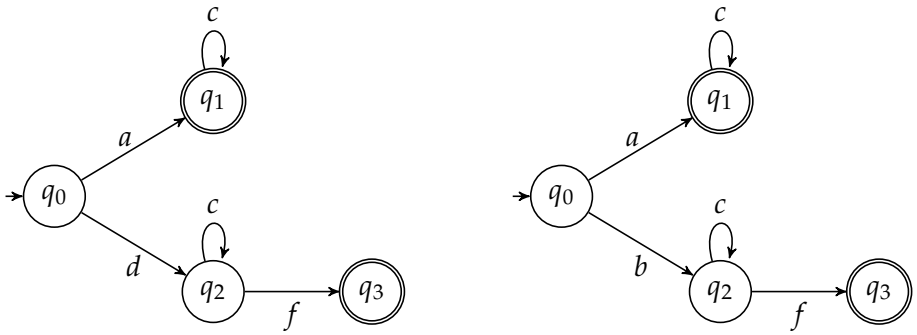
Theorem 16. *Let $\mathcal{D}_{\mathcal{L}}$ be the minimum DFA of $\mathcal{L}$, with initial state q_0 and dimension $n = |D_{\mathcal{L}}|$.*
$\mathcal{L}$ is not Wheeler if and only if there exist μ, ν and γ in Σ^*, with $\gamma \not\prec \mu, \nu$, such that:

1. *$\mu \not\equiv_{\mathcal{L}} \nu$ and they label paths from q_0 to states u and v, respectively;*
2. *γ labels two cycles, one starting from u and one starting from v;*
3. *$\mu, \nu \prec \gamma$ or $\gamma \prec \mu, \nu$.*

The length of the strings μ, ν and γ satisfying the above can be bounded:

4. *$|\mu|, |\nu| \leq |\gamma| \leq n^3 + 2n^2 + n + 2$.*

Since in this work we make an extensive use of Theorem 16, here is a simple example on how and why it works. Consider the two languages $\mathcal{L}_d, \mathcal{L}_b$ recognized by the automaton $\mathcal{A}_d$ on the left and $\mathcal{A}_b$ on the right in Figure 1.5, respectively. As shown in Figure 1.2, the language $\mathcal{L}_d$ is Wheeler, while $\mathcal{L}_b$ is not Wheeler, and this can be easily proved using Theorem 16. In fact, consider the automaton $\mathcal{A}_b$. By setting $\mu := a$, $\nu := b$ and $\gamma := c$, one can verify conditions 1-3 of the theorem are satisfied. Notice that condition $a \not\equiv_{\mathcal{L}_b} b$ follows immediately from the fact that $\delta(q_0, a) \neq \delta(q_0, b)$ in the minimum DFA $\mathcal{A}_b$. If we try to transpose the same reasoning to the automaton $\mathcal{A}_d$ by setting $\mu = a$, $\nu = d$ and $\gamma = c$, condition 3 of Theorem 16 is no longer satisfied. We can not find 3 strings satisfying conditions 1-3 of Theorem 16, confirming that $\mathcal{L}_d$ is Wheeler.



(a) The minimum DFA $\mathcal{D}_{\mathcal{L}_d}$ recognizing $\mathcal{L}_d = ac^* \cup dc^*f$.

(b) The minimum DFA $\mathcal{D}_{\mathcal{L}_b}$ recognizing $\mathcal{L}_b = ac^* \cup bc^*f$.

FIGURE 1.5: The minimum DFAs recognizing the languages $\mathcal{L}_d$ (Wheeler) and $\mathcal{L}_b$ (not Wheeler).

The polynomial bound given by condition 4 of Theorem 16 allows to design an algorithm that decides whether a minimum DFA recognizes a Wheeler language: using dynamic programming (see [30]), it is possible to keep track of all the relevant paths and cycles inside the DFA and check, in polynomial time, whether there exists three strings satisfying the conditions of the theorem. Hence:

Theorem 17 ([32]). *Starting from a DFA recognizing $\mathcal{L}$, the problem of recognizing whether $\mathcal{L}$ is Wheeler is in P.*

Actually, if we want to to decide whether a DFA recognizes a Wheeler language there is an approach way faster than the one previously described: in [33] it was

proved that it is sufficient to compute the minimum DFA $\mathcal{D}_\mathcal{L}$, build the direct product $\mathcal{D}_\mathcal{L} \times \mathcal{D}_\mathcal{L}$ and check for cycles after removing some states. This results in a quadratic algorithm.

As a last note, recall that in the previous section we introduced the class of counter-free automata and star-free languages (see Definition 5). The relation between these two classes and the Wheeler class is stated in the following proposition.

Proposition 18. *Any Wheeler DFA is counter-free. Equivalently, any Wheeler language is star-free.*

Proof. Let $\mathcal{A}$ be a WDFA and suppose, by way of contradiction, that $\mathcal{A}$ has a counter, say $p_0, p_1, \ldots, p_k$ for some $k > 1$ and some string α. Let $<$ be the Wheeler order over the set of states of $\mathcal{A}$ and assume w.l.o.g that $p_{k-1} < p_k$, the case $p_{k-1} > p_k$ being symmetric. Since $\delta(p_{k-2}, \alpha) = p_{k-1}$ and $\delta(p_{k-1}, \alpha) = p_k$ it can be easily proved by induction on the length of α, using condition (W2) of Definition 6, that $p_{k-2} < p_{k-1}$. This argument can be reiterated inductively k times in order to prove that

$$p_k < p_0 < p_1 < \cdots < p_{k-1} < p_k,$$

a contradiction. Hence $\mathcal{A}$ is counter-free. We now prove the second statement of this proposition. Let $\mathcal{L}$ be a Wheeler language. By definition, $\mathcal{L}$ is recognized by a WDFA. From the first statement it follows that WDFA is counter-free, hence the language it recognizes is star-free. □

From the previous proposition, it follows that the presence of a counter in a DFA $\mathcal{D}$ prevents it from being Wheeler. If the given DFA is minimum, this result can be proved directly using Theorem 16: let $p_0, p_1, \ldots, p_k$ be a counter for some $k > 1$ and some string α. Then $\gamma := \alpha^k$ is a string labeling two cycles starting from $u := p_0$ and $v := p_1$. Exploiting the fact that u, v are states belonging to a counter, one can easily find two strings μ, ν reaching u, v such that either $\mu, \nu \prec \gamma$ or $\gamma \prec \mu, \nu$, fulfilling the conditions of Theorem 16. Therefore, the language recognized by $\mathcal{D}$ is not Wheeler, which in turn implies that $\mathcal{D}$ is not Wheeler. This proof can be extended to any DFA —not just the minimum ones— due to the fact that Theorem 16 actually holds for generic DFAs, as we will prove in Lemma 35 in Chapter 2. However, note that the presence of counters is not the only obstacle to an automaton's Wheelerness: as seen in the example in Figure 1.5(b), automata without counters may not be Wheeler. In other words, Wheeler automata are a proper subclass of counter-free automata and, equivalently, Wheeler languages are a proper subclass of star-free languages.

The notation and results listed so far are just a small part of the overall picture regarding the known fact about Wheelerness. For further details, reference is made to works [16] and [23], which serve as the background upon which this thesis is based. One thing to be aware of is that in these two works, two different definitions of Wheelerness are used: the first is more useful when discussing automata, the second is more useful when discussing languages. In this thesis, we will make the same distinction when necessary (see Definition 6 and 11).

Chapter 2

Wheeler complexity

In this chapter we will study some properties related to both Wheeler automata and Wheeler languages. In [16], the first paper that studied Wheeler languages, some problems remained unexplored.

1. What is the complexity of deciding whether a NFA recognizes a Wheeler language?
2. How to compute the minimum WDFA recognizing a language given by its minimum DFA?
3. The notion of Wheelerness depends not only on the structure of an accepting automaton but also on the order of its alphabet. A natural generalization of this notion is the one that drops the request for the alphabet to have a fixed order: if we are allowed to choose any order, what would change? In particular, would we still be able to decide in polynomial time whether exists an order —of the alphabet— that makes a DFA or language Wheeler?
4. In [16] it was shown that there exists a subclass of NFAs, the *reduced* automata, properly containing the DFA class and admitting a representative for any regular languages, for which we can decide whether they are Wheeler in polynomial time (in contrast with the general case [17]). This raises the following questions: how hard it is to decide whether a NFA is reduced? And is it possible to transpose this result to languages, so that we can decide in polynomial time whether there a exists a *reduced* NFA recognizing a Wheeler language?

Before answering to these questions in Sections 2.2, 2.3 and 2.4, in Section 2.1 we will study the state complexity of a couple of operations Wheeler automata are closed for, namely the intersection and the *cascade product* (see Def. 10).

2.1 State complexity

A significant property on the interplay between deterministic and non-deterministic Wheeler Automata is that, given a size-n WNFA $\mathcal{A}$, there always exists a WDFA that recognizes the same language whose size is at most $2n$ (see Theorem 9). The announced amount of states can be computed using the (classic) powerset construction. In other words, the blow-up of the number of states that we might observe when converting NFAs to DFAs, does not occur for Wheeler non-deterministic automata. This property is a direct consequence of an important feature of Wheeler automata: for any state q, the set of strings recognized by q—namely I_q—is a convex set over $\mathrm{Pref}(\mathcal{L})$ with respect to the co-lexicographic order.

State complexity is also used to measure the complexity of operations on regular languages. In this section we prove that the convex property of a Wheeler DFA can

also be exploited to prove that the state complexity of the intersection of Wheeler languages is significantly better than the state complexity of the intersection of general regular languages.

The state complexity of a regular language $\mathcal{L}$ is defined as the number of states of the minimum DFA $\mathcal{D}_{\mathcal{L}}$ recognizing $\mathcal{L}$. The state complexity of an operation on regular languages is a function that associates to the state complexities of the operand languages the worst-case state complexity of the language resulting from the operation. For instance, the state complexity of the intersection of $\mathcal{L}_1$ and $\mathcal{L}_2$ is mn, where m and n are the number of states of $\mathcal{D}_{\mathcal{L}_1}$ and $\mathcal{D}_{\mathcal{L}_2}$ respectively. The bound mn for the intersection can easily be proved using the state-product construction for $\mathcal{D}_{\mathcal{L}_1}$ and $\mathcal{D}_{\mathcal{L}_2}$, and it is a known fact that this bound is tight [34].

It is natural to define the Wheeler state complexity of a Wheeler language $\mathcal{L}$ as the number of states of the minimum WDFA $\mathcal{D}^W_{\mathcal{L}}$ recognizing $\mathcal{L}$. In the following theorem, we show what it is the Wheeler state complexity of the intersection of two Wheeler languages $\mathcal{L}_1$ and $\mathcal{L}_2$.

Lemma 19. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two WDFAs recognizing the languages $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. Then, the direct product $\mathcal{D} := \mathcal{D}_1 \times \mathcal{D}_2$ recognizing the language $\mathcal{L} := \mathcal{L}_1 \cap \mathcal{L}_2$ is Wheeler and it has at most $|D_1| + |D_2| - |\Sigma| - 1$ states.*

Proof. First we prove that $|D_1| + |D_2| - |\Sigma| - 1$ is an upper bound to the number of states of $\mathcal{D}$. In general, for DFAs, the following holds: let p be a state of a DFA $\mathcal{D}'_1$ and let I_p be its input language. Let q be a state of a DFA $\mathcal{D}'_2$ and let I_q be its input language. Then, the input language of the state (p, q) of the direct product $\mathcal{D}'_1 \times \mathcal{D}'_2$ is

$$I_{(p,q)} = I_p \cap I_q.$$

In particular, if $I_p \cap I_q = \emptyset$ then the state (p, q) has an empty incoming language and hence it is unreachable. It immediately follows that the number of reachable states of $\mathcal{D}'_1 \times \mathcal{D}'_2$ is at most equal to the number of non-empty intersection of the form $I_p \cap I_q$. In our case, $\mathcal{D}_1$ is Wheeler, so let $p_0 < p_1 \cdots < p_{n-1}$ be the Wheeler order among the states of $\mathcal{D}_1$. From Proposition 5 it follows that the input languages are ordered as follows:

$$I_{p_0} \prec_{ps} I_{p_1} \prec_{ps} \cdots \prec_{ps} I_{p_{n-1}}.$$

Similarly, let the input languages of the states of $\mathcal{D}_2$ be ordered as

$$I_{q_0} \prec_{ps} I_{q_1} \prec_{ps} \cdots \prec_{ps} I_{q_{m-1}}.$$

To obtain an upper bound to the number of states of $\mathcal{D}$ is then sufficient to count the number of non-empty intersections of the form $I_{p_i} \cap I_{q_j}$, for $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$. Due to input-consistency, all strings belonging to the input language of a given state of $\mathcal{D}_1$ or $\mathcal{D}_2$ end with the same character. Languages I_{p_i} and I_{q_j} that end with different characters of the alphabet must have empty intersection, hence we will focus only on input languages whose elements end with a specific character, say a. Notice that we need to treat separately the languages $I_{p_0} = I_{q_0} = \{\varepsilon\}$, which always lead to the non-empty intersection $I_{p_0} \cap I_{q_0} = \{\varepsilon\}$.

Let $I_{p^a_1}, \ldots, I_{p^a_{n_a}}$ be the input languages of $\mathcal{D}_1$ that end with a and let $I_{q^a_1}, \ldots, I_{q^a_{m_a}}$ be the input languages of $\mathcal{D}_2$ that end with a. We suppose both lists are ordered by $\prec_{ps}$. Let k be the number of non-empty intersections of the form $I_{p^a_i} \cap I_{q^a_j}$, and let $\alpha_1 \prec \cdots \prec \alpha_k$ be an ordered list containing one representatives for each non-empty intersection. For any $1 \leq s < k$, consider the strings α_s and α_{s+1}. There must exist four unique indexes i, j, i', j' such that $\alpha_s \in I_{p^a_i} \cap I_{q^a_j}$ and $\alpha_{s+1} \in I_{p^a_{i'}} \cap I_{p^a_{j'}}$. From

$\alpha_s \prec \alpha_{s+1}$ it follows that both $i \leq i'$ and $j \leq j'$ hold, since the input languages of both $\mathcal{D}_1$ and $\mathcal{D}_2$ are pairwise disjoint and ordered by $\preceq_{ps}$. On the other hand, it can not be the case that both $i = i'$ and $j = j'$ hold, because α_s and α_{s+1} belong to different intersections. Therefore we have that $i' + j' \geq i + j + 1$. The values of the function $f(\alpha_s) = i + j$ can range from 2 to $n_a + m_a$, hence there might be at most $n_a + m_a - 1$ different representatives. Taking the sum over every possible characters of Σ and adding the class $W_0^1 \cap W_0^2 = \{\varepsilon\}$, we get an upper bound of

$$1 + \sum_{a \in \Sigma} (n_a + m_a - 1) = 1 + \sum_{a \in \Sigma} n_a + \sum_{a \in \Sigma} m_a - |\Sigma| = \\ = 1 + (n-1) + (m-1) - |\Sigma| = n + m - |\Sigma| - 1$$

different possible representatives.

Second, we prove that $\mathcal{D}$ is Wheeler. From Proposition 6 it follows that $\mathcal{D}$ is Wheeler iff the order $<_{\mathcal{D}}$ is total. Therefore, it it sufficient to prove that given two distinct —reachable— states $(p, q), (p', q')$ of $\mathcal{D}$, either $(p, q) <_{\mathcal{D}} (p', q')$ or $(p', q') <_{\mathcal{D}} (p, q)$ holds. Hence, let $(p, q), (p', q')$ be two distinct states of $\mathcal{D}$ such that $I_p \cap I_q \neq \emptyset \neq I_{p'} \cap I_{q'}$ and assume, by way of contradiction, that $(p, q), (p', q')$ are incomparable. It can not be the case that both $I_p \preceq_{ps} I_{p'}$ and $I_q \preceq_{ps} I_{q'}$ hold (with at least one inequality being strict since $(p, q), (p', q')$ are distinct), otherwise we would have $I_{(p,q)} \prec_{ps} I_{(p',q')}$ and hence $(p, q) <_{\mathcal{D}} (p', q')$, a contradiction. Similarly, it can not be the case that both $I_{p'} \preceq_{ps} I_p$ and $I_{q'} \preceq_{ps} I_q$ hold. Therefore, consider the case where $I_p \prec_{ps} I_{p'}$ and $I_{q'} \prec_{ps} I_q$ (the other case can be treated symmetrically), and let α, β be two strings in $I_p \cap I_q$ and $I_{p'} \cap I_{q'}$ respectively. From $I_p \prec_{ps} I_{p'}$ it follows that $\alpha \prec \beta$, whereas from $I_{q'} \prec_{ps} I_q$ it follows that $\beta \prec \alpha$, a contradiction. Therefore the order $<_{\mathcal{D}}$ is total and $\mathcal{D}$ is Wheeler. □

Similarly to the case of determinizing a WNFA, where we used the classic powerset construction without generating "too many" states, the classic direct-product construction computes a WDFA that recognizes the intersection of the languages accepted by two WDFAs $\mathcal{W}_1$ and $\mathcal{W}_2$ without producing non-necessary states. This time, the number of states generated will be at most the sum of the number of states of $\mathcal{W}_1$ and $\mathcal{W}_2$. Moreover, we can use Lemma 19 to analyze the state complexity of the operation intersection over WDFAs, as stated in the following corollary.

Corollary 19.1. *Let $\mathcal{D}^W_{\mathcal{L}_1}$ and $\mathcal{D}^W_{\mathcal{L}_2}$ be the minimum WDFAs recognizing the languages $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. Then, the minimum WDFA recognizing $\mathcal{L} := \mathcal{L}_1 \cap \mathcal{L}_2$ has at most $|D^W_{\mathcal{L}_1}| + |D^W_{\mathcal{L}_2}| - |\Sigma| - 1$ states.*
This bound is tight.

Proof. We apply Lemma 19 to the WDFAs $\mathcal{D}^W_{\mathcal{L}_1}$ and $\mathcal{D}^W_{\mathcal{L}_2}$ to derive that their direct product is a WDFA with at most $|D^W_{\mathcal{L}_1}| + |D^W_{\mathcal{L}_2}| - |\Sigma| - 1$ states that recognizes $\mathcal{L}$. The thesis immediately follows.

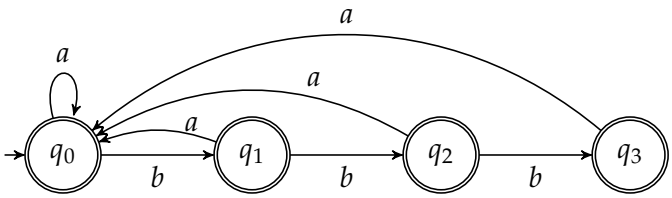
To show that the bound is tight (at least for $|\Sigma| = 2$), consider the following families of languages over the alphabet $\Sigma = \{a, b\}$, with $a \prec b$:

$$\mathcal{A}_n := \{\alpha \in \Sigma^* : \ a^{n+1} \text{ is not a factor of } \alpha\}$$
$$\mathcal{B}_m := \{\beta \in \Sigma^* : \ b^{m+1} \text{ is not a factor of } \beta\}.$$

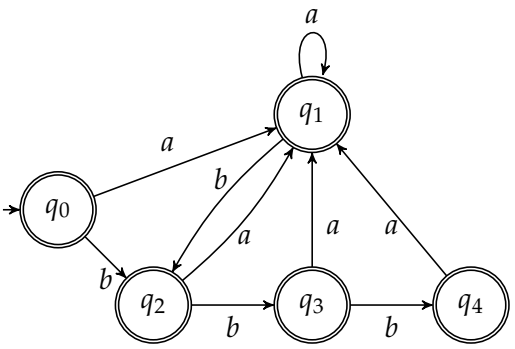
We can easily prove that all these languages are Wheeler. The minimum DFA recognizing $\mathcal{B}_m$, see Figure 2.1, is almost a WDFA: the only unmet condition is the requirement that the initial state has no incoming edges. Therefore, it is sufficient to split

the state q_0, resulting in an automaton with $m+2$ states. A list of representatives of the $\equiv^c_{\mathcal{B}_m}$-classes is

$$\varepsilon, a, b, \ldots, b^m.$$



(a) The minimum DFA $\mathcal{D}_{\mathcal{B}_3}$.



(b) The minimum WDFA $\mathcal{D}^W_{\mathcal{B}_3}$.

FIGURE 2.1: The minimum DFA and the minimum WDFA recognizing the language $\mathcal{B}_3$.

The minimum WDFA recognizing $\mathcal{A}_n$ has more states than the minimum DFA: for $1 \leq i < n$ we have that a^i and ba^i belongs to the same MN-class, which does not contain a^n. Since $a^i \prec a^n \prec ba^i$, we have to split the $\equiv_{\mathcal{A}_n}$-class containing both a^i and ba^i into two different $\equiv^c_{\mathcal{A}_n}$-classes. The automaton has $2n+1$ states, see Figure 2.2. A list of representatives of the $\equiv^c_{\mathcal{A}_n}$-classes is

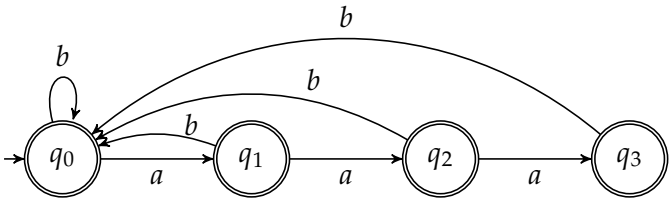
$$\varepsilon, a, \ldots, a^n, ba^{n-1}, \ldots, ba, b.$$

We already proved that the language $\mathcal{L} := \mathcal{A}_n \cap \mathcal{B}_m$ might have at most $(2n+1)+(m+2)-|\Sigma|-1 = 2n+m$ different $\equiv^c_{\mathcal{L}}$-classes, hence it is sufficient to show that there are at least $2n+m$ different ones. We claim that the $2n+m$ strings

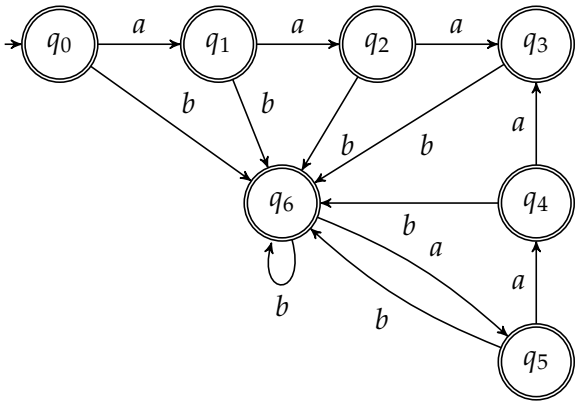
$$\varepsilon, a, \ldots, a^n, ba^{n-1}, \ldots, ba, b, \ldots, b^m$$

all belong to different $\equiv^c_{\mathcal{L}}$-classes. Strings that end with a different amount of a's (or b's) belong to different $\equiv_{\mathcal{L}}$-classes, so there is nothing to prove. Therefore we only have to check, for each $1 \leq i < n$, that a^i and ba^i belong to different $\equiv^c_{\mathcal{L}}$-classes, and again this is true since $a^i \prec a^n \prec ba^i$. □

As we proved in Lemma 19, if we are given two WDFAs and we compute their direct product we obtain a DFA that is still Wheeler. It is natural to ask whether there are more operations preserving the Wheeler properties. As far as the classic operations on DFAs are concerned (booleans, concatenation, and Kleene star), the answer is no: in [16] it was proved that intersection (that is, the direct product) is the only "classic" automata operation for which Wheeler automata are closed in



(a) The minimum DFA $\mathcal{D}_{\mathcal{A}_3}$.



(b) The minimum WDFA $\mathcal{D}^W_{\mathcal{A}_3}$.

FIGURE 2.2: The minimum DFA and the minimum WDFA recognizing the language $\mathcal{A}_3$.

full generality. In addition, we will show now that there exists one more operation preserving Wheelerness, which can be considered as a generalization of the direct product of two automata.

One can picture the direct product of two DFAs $\mathcal{D}_1 = (Q^1, \delta^1, \dots)$ and $\mathcal{D}_2 = (Q^2, \delta^2, \dots)$ rather simply as follows: starting from the (usually excessively large) set of states $Q^1 \times Q^2$, in order to determine the existence of an edge $((q, r), a, (q', r'))$, check whether the two edges (q, a, q') and (r, a, r') were present in the original DFAs.

This construction can be generalized to obtain the *cascade product* (see [25]). The idea is to combine the two DFAs with the first one working as usual (receiving a string as an input), while the second one takes as input both the string and the run of the first automaton over this string, as depicted in Figure 2.3.

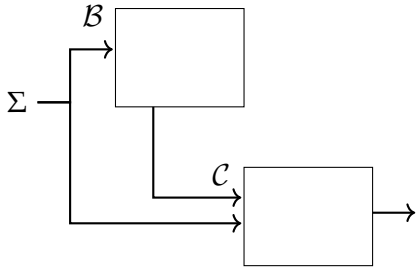


FIGURE 2.3: The automata $\mathcal{B}$ and $\mathcal{C}$. The two arrows from Σ to $\mathcal{C}$ and from $\mathcal{B}$ to $\mathcal{C}$ indicate that, at each step, the automaton $\mathcal{C}$ reads both the input letter from Σ and the current state of the automaton $\mathcal{B}$.

Since the second automaton receives as input both the output —the run— of the first automaton on a string and the string at the same time, its alphabet must be made

of pairs: the first component of such a pair is a state of the first automaton, whereas the second component must be a character of the alphabet Σ of the first automaton. Formally, we define the *cascade product* as follows.

Definition 10 (Cascade product)**.** Let $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, \delta^{\mathcal{B}})$ be DFA and $\mathcal{C} = (Q^{\mathcal{C}}, Q^{\mathcal{B}} \times \Sigma, \delta^{\mathcal{C}})$ be a second DFA whose alphabet is the cartesian product of $Q^{\mathcal{B}}$ and Σ. The *cascade product* $\mathcal{B} \circ \mathcal{C} = (Q^{\mathcal{B}\circ\mathcal{C}}, \Sigma, \delta^{\mathcal{B}\circ\mathcal{C}})$ is the automaton with set of states $Q^{\mathcal{B}\circ\mathcal{C}} := Q^{\mathcal{B}} \times Q^{\mathcal{C}}$ and transition function defined by

$$\delta^{\mathcal{B}\circ\mathcal{C}}\big((q,r),a\big) = \big(\delta^{\mathcal{B}}(q,a), \delta^{\mathcal{C}}(r,(q,a))\big).$$

Clearly the cascade product, defined for DFAs, is still a DFA. Note that both $\delta^{\mathcal{B}}$ and $\delta^{\mathcal{C}}$ might be partial functions, thus we are implicitly requiring that $\delta^{\mathcal{B}\circ\mathcal{C}}((q,r),a) \neq \bot$ if and only if $\delta^{\mathcal{B}}(q,a) \neq \bot \ \wedge \ \delta^{\mathcal{C}}(r,(q,a)) \neq \bot$.

Figure 2.4 shows an example of cascade product between two automata.

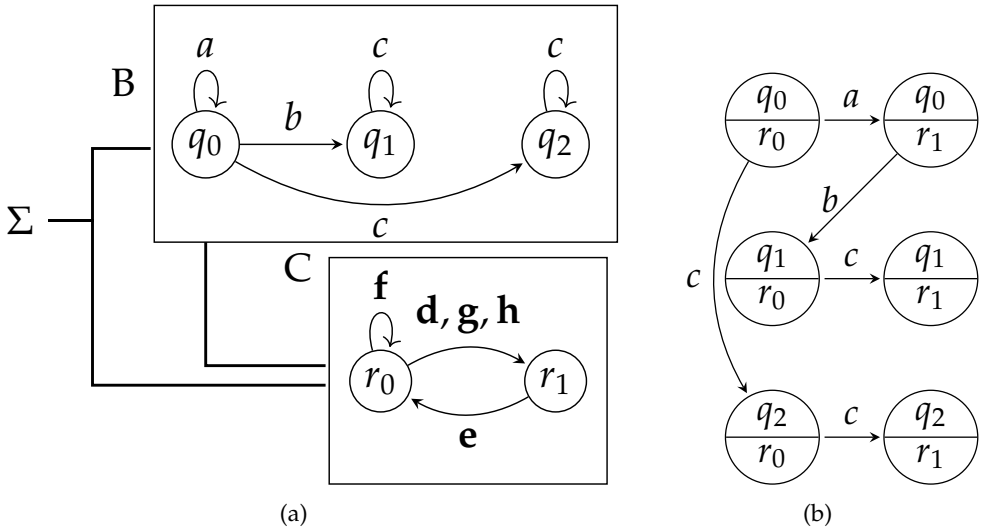


FIGURE 2.4: In 2.4(a), the automata $\mathcal{B}$ and $\mathcal{C}$. $\mathcal{C}$-transitions are: $\mathbf{d} = (q_0, a), \mathbf{e} = (q_0, b), \mathbf{f} = (q_0, c), \mathbf{g} = (q_1, c), \mathbf{h} = (q_2, c)$. The links from Σ and $\mathcal{B}$ to $\mathcal{C}$ indicate that, at each step, $\mathcal{C}$ reads both the input letter from Σ and $\mathcal{B}$'s current state. In 2.4(b), the cascade product $\mathcal{B} \circ \mathcal{C}$ of $\mathcal{B}$ and $\mathcal{C}$.

Notice that the direct product between automata can be seen as a particular case of the cascade product: given two automata $\mathcal{B}, \mathcal{C}$ the direct product $\mathcal{B} \times \mathcal{C}$ is equal to the cascade product $\mathcal{B} \circ \mathcal{C}'$, where $\mathcal{C}'$ is obtained from $\mathcal{C}$ by replacing each edge (r, a, r') with $|S^B|$ edges $\big(r, (q,a), r'\big)$, one for each $q \in S^B$. This way, we have $\delta^{\mathcal{C}'}(r,(q,a)) = r' = \delta^{\mathcal{C}}(r,a)$, therefore the cascade product operates as follows:

$$\delta^{\mathcal{B}\circ\mathcal{C}'}\big((q,r),a\big) = \big(\delta^{\mathcal{B}}(q,a), \delta^{\mathcal{C}'}(r,(q,a))\big) = \big(\delta^{\mathcal{B}}(q,a), \delta^{\mathcal{C}}(r,a)\big);$$

this definition coincides with the one of $\delta^{\mathcal{B}\times\mathcal{C}}$.

We will make good use of this new operation in Section 3, but for now we will just show a nice property preserved by it: as it holds for the direct product, Wheeler automata are closed under the cascade product and the state complexity of this operation is linear in the Wheeler case. Actually, we will prove this result for a class of automata slightly larger than Wheeler automata, that is, a class that avoids the input-consistency requirement. Working with input-consistent automata might be

cumbersome due to the larger amount of states, but we can eliminate this restriction by giving an alternative, less strict definition of Wheeler automata as given in [23]. The new class of automata is a proper superclass of the old one but the class of recognized languages is exactly the Wheeler class. For this reason, we still refer to these automata as Wheeler automata.

Definition 11. Let $\mathcal{D} = (Q, q_0, \delta, F)$ be a DFA. We say that $\mathcal{D}$ is Wheeler if there exists a total order $<$ on Q having the initial state q_0 as minimum that satisfies the following two axioms:

(W1′) For every $u, v \in Q$, if $u < v$, then, if $u = \delta(u', a)$, $v = \delta(v', b)$, it holds $a \preceq b$;

(W2′) For every $a \in \Sigma$, if $u = \delta(u', a)$, $v = \delta(v', a)$, and $u < v$, then $u' < v'$ (it must be $u' \neq v'$ because u and v are distinct).

Notice that, since we are requiring that $<$ is a total order, axiom (W2′) is equivalent to: for every $a \in \Sigma$, if $u = \delta(u', a)$, $v = \delta(v', a)$, and $u' < v'$, then $u \leq v$. We will freely use both versions in the following. This alternative definition of Wheeler automata is useful not only because it allows us to work with non-input-consistent automata but also because it allows us to generalize the notion of Wheelerness to any automaton: if we allow the order on states to be partial rather than total, the new definition fits generic NFAs. This way, we can introduce a measure, called the *width* of the automaton, which indicates how far the partial order is from being total, or how far the automaton is from being Wheeler. We will explore this topic in more detail in Chapter 4.

It can easily be proved that a Wheeler automaton (accordingly to Definition 11) recognizes a Wheeler language by transforming it into an input-consistent automaton: this new automaton is Wheeler accordingly to Definition 6. For this reason, we will freely switch between this two definitions. In particular, we will use Definition 11 for the following lemma and in Section 3 and Section 4.

Lemma 20. *If $B = (Q, \Sigma, q_0, \delta_B)$ is Wheeler with respect to the orders $(\Sigma, \prec)$ and $(Q, <_Q)$ and $C = (R, Q \times \Sigma, r_0, \delta_C)$ is Wheeler with respect to the co-lexicographic order on the set of pairs[1] $Q \times \Sigma$ and the order $(R, <_R)$ then $B \circ C$ (restricted to accessible states) is a WDFA with respect to $(\Sigma, \prec)$ and the lexicographic order over $Q \times R$.*

Moreover, the lexicographic order on $Q \times R$-accessible states coincides with the co-lexicographic order and if B has n_1 states and C has n_2 states then the cascade product $B \circ C$ has at most $n_1 + n_2 - 1$ states.

Proof. We use the symbols $\preceq$ ($\prec$) to denote both the (strict) order on Σ and the (strict) co-lexicographic order on $(Q \times \Sigma)$. If u is an automaton state, $\lambda(u)$ denote the set of characters labeling transitions arriving in u. Let $\leq$ ($<$) denotes the (strict) lexicographic order over $(Q, <_Q) \times (R, <_R)$. Following Definition 11, we have to prove that:

[W1′] if $\delta^{B\circ C}((q_1', r_1'), a) = (q_1, r_1)$ and $\delta^{B\circ C}((q_2', r_2'), b) = (q_2, r_2)$, with $(q_1, r_1) < (q_2, r_2)$, then $a \preceq b$;

[W2′] if $(q_1, r_1) < (q_2, r_2)$ then $\delta^{B\circ C}((q_1, r_1), a) \leq \delta^{B\circ C}((q_2, r_2), a)$, for all $a \in \Sigma$.

[W1′]: From the hypothesis we know that

$$\delta^{B\circ C}((q_1', r_1'), a) = \left(\delta^B(q_1', a), \delta^C(r_1', (q_1', a))\right) = (q_1, r_1)$$

[1] That is: $(q, a) \prec (q', b) \Leftrightarrow (a \prec b) \vee [(a = b) \wedge (q <_Q q')]$ and dually for the lexicographic order over pairs.

and

$$\delta^{B\circ C}\big((q_2', r_2'), b\big) = \Big(\delta^B(q_2', b), \delta^C\big(r_2', (q_2', b)\big)\Big) = (q_2, r_2).$$

Suppose $q_1 <_Q q_2$. Then, since $a \in \lambda(q_1)$ and $b \in \lambda(q_2)$ and B is Wheeler, we have $a \preceq b$. If instead $q_1 = q_2$ then $r_1 <_R r_2$. Then, since $(q_1', a) \in \lambda(r_1)$ and $(q_2', b) \in \lambda(r_2)$ and C is Wheeler, we have $(q_1', a) \preceq (q_2', b)$ in the co-lexicographic order on $Q \times \Sigma$, implying $a \preceq b$.

[W2']: Suppose $(q_1, r_1) < (q_2, r_2)$ and let

$$(q_1', r_1') = \delta^{B\circ C}\big((q_1, r_1), a\big) = \Big(\delta^B(q_1, a), \delta^C\big(r_1, (q_1, a)\big)\Big)$$

and

$$(q_2', r_2') = \delta^{B\circ C}\big((q_2, r_2), a\big) = \Big(\delta^B(q_2, a), \delta^C\big(r_2, (q_2, a)\big)\Big)$$

We have to prove that $(q_1', r_1') \leq (q_2', r_2')$. If $q_1 <_Q q_2$ then $q_1' = \delta^B(q_1, a) \leq q_2' = \delta^B(q_2, a)$. Moreover, from $q_1 <_Q q_2$ we have $(q_1, a) \prec (q_2, a)$ hence $r_1' = \delta^C(r_1, (q_1, a)) \leq_R r_2' = \delta^C(r_2, (q_2, a))$. This proves that $(q_1', r_1') \leq (q_2', r_2')$ in the case $q_1 <_Q q_2$.

If $q_1 = q_2$ then $q_1' = \delta^B(q_1, a) = \delta^B(q_2, a) = q_2'$ and $r_1 < r_2$ holds. In this case $r_1' = \delta^C(r_1, (q_1, a)) \leq_R \delta^C(r_2, (q_1, a)) = \delta^C(r_2, (q_2, a)) = r_2'$. Hence $r_1' \leq_R r_2'$ and, since $q_1' = q_2'$ this proves $(q_1', r_1') \leq (q_2', r_2')$.

We now prove that the number of reachable states in $B \circ C$ is at most $n_1 + n_2 - 1$. Notice that a state (q, r) in $B \circ C$ is reachable iff

$$I_{(q,r)} = \{\alpha \in \Sigma^* \,:\, (q, r) \in \delta^{B\circ C}((q_0, r_0), \alpha)\} \neq \emptyset.$$

In the following, if ν is a string (of states, or letters) we denote by $\nu[i]$ the leftmost i-th element of the string and by $\nu^-[i]$ the rightmost i-th element of the string.

Notice that, if $|\alpha| = k$, then

$$\alpha \in I_{(q,r)} \;\Leftrightarrow\; \alpha \in I_q \text{ and } (\pi_\alpha[1], \alpha[1]) \dots (\pi_\alpha[k], \alpha[k]) \in I_r,$$

where $\pi_\alpha[1] = q_0$ and π_α is the B-computation made by B on reading α.

We first claim that, given α, β with $|\alpha| = k$ and $|\beta| = h$ and corresponding B-computations π_α, π_β then

$$(\pi_\alpha[1], \alpha[1]) \dots (\pi_\alpha[k], \alpha[k]) \preceq (\pi_\beta[1], \beta[1]) \dots (\pi_\beta[h], \beta[h]) \Leftrightarrow (\pi_\alpha, \alpha) \preceq (\pi_\beta, \beta) \tag{2.1}$$

(where the orders appearing are both colex orders: on the left the colex ordering of string in the alphabet $Q \times \Sigma$, on the right the colex order of pairs of strings (Q^*, Σ^*), where the strings in Q^*, Σ^* are co-lexicographically ordered). Notice that this property does not hold for general DFAs, but only because B and C are Wheeler.

We may suppose that $\alpha \prec \beta$: if $\alpha = \beta$ then $\pi_\alpha = \pi_\beta$ and the result follows; if $\beta \prec \alpha$ we can proceed symmetrically.

Suppose $\alpha = \alpha'\gamma$ and $\beta = \beta'\gamma$ with $\alpha' \prec \beta'$ ending in different letters (or $\alpha' = \epsilon$). Since we are dealing with colex orders, then the equivalence 2.1 holds for α' and β'.

If $|\gamma| = n$, we prove that

$$\pi_\alpha^-[n] \preceq \pi_\beta^-[n], \dots, \pi_\alpha^-[1] \preceq \pi_\beta^-[1] \tag{2.2}$$

(where α' ends in $\pi_\alpha^-[n]$ and β' ends in $\pi_\beta^-[n]$).

Since $\alpha' \prec \beta'$ and α', β' end respectively in $\pi_\alpha^-[n]$, $\pi_\beta^-[n]$, we have $\pi_\alpha^-[n] \leq_Q \pi_\beta^-[n]$ in the B-Wheeler order $\leq_Q$. If $\pi_\alpha^-[n] = \pi_\beta^-[n]$ then $\pi_\alpha^-[i] = \pi_\beta^-[i]$, for $i = 1, \ldots, n$ because B is a DFA. If $\pi_\alpha^-[n] < \pi_\beta^-[n]$ then, since

$$\pi_\alpha^-[n-1] = \delta^B(\pi_\alpha^-[n], \gamma[1]), \pi_\beta^-[n-1] = \delta^B(\pi_\beta^-[n], \gamma[1])$$

we have $\pi_\alpha^-[n-1] \leq \pi_\beta^-[n-1]$, and so forth, proving that $\pi_\alpha^-[i] \leq \pi_\beta^-[i]$ for all i.

From the validity of the equivalence 2.1 for α' and β' and 2.2 we easily obtain 2.1 for α and β. Notice that 2.1 means that we can identify strings over the alphabet $Q \times \Sigma$ and the colex order between these strings, that is, C-inputs, with colexicographically ordered pairs in (Q^*, Σ^*).

We now claim that

$$I_{(q,r)} \neq \emptyset \wedge I_{(q',r')} \neq \emptyset \wedge (q,r) <_{lex} (q',r') \wedge q < q' \Rightarrow r \leq r' \tag{2.3}$$

(where $<_{lex}$ in the lexicographic order of the states in $B \circ C$). Suppose $\alpha \in I_{(q,r)}$, $\beta \in I_{(q',r')}$. Then $\alpha \in I_q, (\pi_\alpha, \alpha) \in I_r$, $\beta \in I_{q'}, (\pi_\beta, \beta) \in I_{r'}$.

If $q <_Q q'$ then, since B is Wheeler, we have $\alpha \prec \beta$. Then we have $(\pi_\alpha, \alpha) \prec (\pi_\beta, \beta)$; since $(\pi_\alpha, \alpha) \in I_r$ and $(\pi_\beta, \beta) \in I_{r'}$ and C is Wheeler, this implies $r \leq r'$.

From the claim the bound $n_1 + n_2 - 1$ follows. Moreover, from 2.3 it also holds that

$$(q,r) <_{lex} (q',r') \Leftrightarrow (q,r) <_{colex} (q',r').$$

□

Remark 21. Notice that the upper bound in Lemma 20 is worse than the bound in Lemma 19. This is to be expected since we are using Definition 11 instead of Definition 6: the latter allows —in general— for smaller automata recognizing a given language. Using Definition 11 in Lemma 19 would result in the same upper bound.

2.2 Computing the minimum WDFA

Despite the good behaviour that Wheeler automata show regarding determinization, cascade products, and intersection, there are cases when the state complexity of a construction turns out exponential. In fact, it is known [30, 33, 35] that a blow-up of states can occur when switching from the minimum DFA recognizing a language $\mathcal{L}$ to its minimum WDFA. In this section we provide an algorithm to compute the minimum WDFA $\mathcal{D}_\mathcal{L}^W$ starting from the minimum DFA $\mathcal{D}_\mathcal{L}$ of a Wheeler language $\mathcal{L}$ and from a *$\mathcal{L}$-fingerprint*, that is, a set of strings containing exactly one representative of each $\equiv_\mathcal{L}^c$-class of $\mathcal{L}$. Then we describe how to extract a fingerprint of $\mathcal{L}$ starting from $\mathcal{D}_\mathcal{L}$.

Definition 12 (Fingerprint)**.** Let $\mathcal{L}$ be a Wheeler language, and let m be the number of equivalence classes of $\equiv_\mathcal{L}^c$. A set of strings $F = \{\alpha_1, \ldots, \alpha_m\} \subseteq \Sigma^*$ is called a *fingerprint* of $\mathcal{L}$ if and only if for each $\equiv_\mathcal{L}^c$-class C it holds $|F \cap C| = 1$.

We start by proving the existence of an upper bound to the length of the elements of a fingerprint.

Lemma 22. *Let $\mathcal{D}_\mathcal{L}$ be the minimum DFA recognizing the Wheeler language $\mathcal{L}$ over the alphabet Σ and let $W_1, ..., W_m$ be the pairwise distinct Wheeler equivalence classes of $\equiv_\mathcal{L}^c$. Then, for each $1 \leq i \leq m$, there exists a string $\alpha_i \in W_i$ such that $|\alpha_i| < n + n^2$, where $n := |\mathcal{D}_\mathcal{L}|$.*

Proof. Suppose, reasoning by contradiction, that there exists a class W_i such that for all $\alpha \in W_i$ it holds $|\alpha| \geq n + n^2$ and let $\alpha \in W_i$ be a string of minimum length. We use the minimality of α to find three strings μ, ν, γ that meet the requirements of Theorem 16, contradicting the hypothesis that $\mathcal{L}$ is Wheeler; to do so, we analyze the computation of $\mathcal{D}_\mathcal{L}$ over α. Consider the first $n+1$ states $q_0 = t_0, ..., t_n$ of $\mathcal{D}_\mathcal{L}$ visited by reading the first n characters of α. Since $\mathcal{D}_\mathcal{L}$ has only n states, there must exist $0 \leq i, j \leq n$ with $i < j$ such that $t_i = t_j$. Let α' be the prefix of α of length i (if $i = 0$ then $\alpha' = \varepsilon$), let $\delta \neq \varepsilon$ be the factor of α of length $j - i$ labeling the cycle $t_i, ..., t_j$, and let ζ be the suffix of α such that $\alpha = \alpha'\delta\zeta$, as depicted in Figure 2.5. Notice that $\zeta \neq \epsilon$ since $|\alpha| \geq n + n^2$. By construction, the strings α and $\beta := \alpha'\zeta$ end in the same state, hence $\alpha \equiv_\mathcal{L} \beta$. Moreover, from $|\beta| < |\alpha|$ and the minimality of α in its class it follows that $\alpha \not\equiv^c_\mathcal{L} \beta$.
Suppose that $\alpha \prec \beta$, the other case being completely symmetrical. Since α and β share the same suffix $\zeta \neq \varepsilon$, they end with the same character. Moreover, since the strings α and β are Myhill-Nerode equivalent but not $\equiv^c_\mathcal{L}$ equivalent there must exists a string η such that $\alpha \prec \eta \prec \beta$ and $\eta \not\equiv_\mathcal{L} \alpha$ (see Def.9).
From $\zeta \dashv \alpha, \beta$ it follows that $\zeta \dashv \eta$, so we can write $\eta = \eta'\zeta$ for some $\eta' \in \Sigma^*$. Recall that by construction $\alpha = \alpha'\delta\zeta$ with $|\alpha'\delta| \leq n$, hence $|\zeta| \geq n^2$. Consider the last $n^2 + 1$ states $r_0, ..., r_{n^2}$ of $\mathcal{D}_\mathcal{L}$ visited by reading the string α and the last $n^2 + 1$ states $p_0, ..., p_{n^2}$ visited by reading the string η (these two paths have the same label which is a suffix of ζ). Since $\mathcal{D}_\mathcal{L}$ has only n states, there must exist $0 \leq i, j \leq n^2$ with $i < j$ such that $(r_i, p_i) = (r_j, p_j)$. Notice that it can't be $r_i = p_i$, otherwise from the determinism of $\mathcal{D}_\mathcal{L}$ it would follow $r_{n^2} = p_{n^2}$; from the minimality of $\mathcal{D}_\mathcal{L}$ it would then follow $\alpha \equiv_\mathcal{L} \eta$, a contradiction.
Let ζ'' be the suffix of ζ of length $n^2 - j$, and let γ be the factor of ζ of length $j - i$ labeling the cycles $r_i, ..., r_j$ and $p_i, ..., p_j$. Since $|\zeta| \geq n^2$, there exists $\zeta' \in \Sigma^*$ such that $\zeta = \zeta'\gamma\zeta''$. We can then rewrite α, η and β as

$$\begin{aligned}\alpha &= \alpha'\delta\zeta = \alpha'\delta\zeta'\gamma\zeta'' \\ \eta &= \eta'\zeta = \eta'\zeta'\gamma\zeta'' \\ \beta &= \alpha'\zeta = \alpha'\zeta'\gamma\zeta'',\end{aligned}$$

where, in $\mathcal{D}_\mathcal{L}$, the strings $\alpha'\zeta'$ and $\alpha'\delta\zeta'$ both reach r_i whereas the string $\eta'\zeta'$ reaches p_i. Let k be an integer such that $|\gamma^k|$ is greater than $|\alpha'\delta\zeta'|$ and $|\eta'\zeta'|$. Set $\mu := \eta'\zeta'$; from $\alpha \prec \eta \prec \beta$ it follows that $\alpha'\delta\zeta' \prec \mu \prec \alpha'\zeta'$. If $\gamma^k \prec \mu$ set $\nu := \alpha'\zeta'$, otherwise set $\nu := \alpha'\delta\zeta'$. In both cases, the hypothesis of Theorem 16 are satisfied, since γ^k labels two cycles starting from the states r_i and p_i, that we have proved to be distinct. We can conclude that $\mathcal{L}$ is not Wheeler, a contradiction, and the thesis follows. □

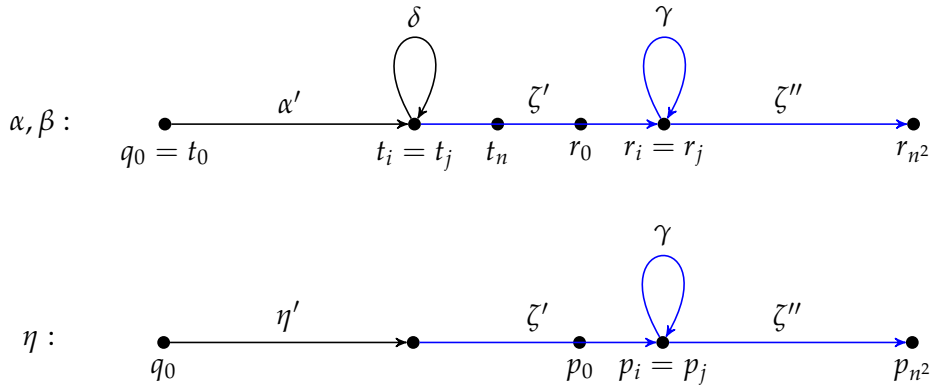


FIGURE 2.5: Strings α, β, η and the significant states they pass through. The string $\zeta = \zeta'\gamma\zeta''$ is highlighted in blue.

The upper bound of Lemma 22 was improved in [35] from $n^2 + n$ to $2n$. This improvement reflects on all the results we will show in this section. Therefore, we will present all the results using the parameter value UB which, as demonstrated so far, is equal to $n^2 + n$. It is worth keeping in mind that this value can be replaced with $2n$.

We show now how to compute the minimum WDFA recognizing a Wheeler language $\mathcal{L}$ if we are given its minimum DFA $\mathcal{D}_{\mathcal{L}}$ and one of its fingerprints.

Proposition 23 (Fingerprint to min WDFA)**.** *Let $\mathcal{D}_{\mathcal{L}}$ be the minimum automaton recognizing the Wheeler language $\mathcal{L}$ with $|\mathcal{D}_{\mathcal{L}}| = n$, alphabet Σ with $|\Sigma| = \sigma$, and let $W_1, ..., W_m$ be the pairwise distinct equivalence classes of $\equiv^c_{\mathcal{L}}$. Assume that we are given a* fingerprint *of $\mathcal{L}$, whose elements have length less than $UB = n^2 + n$. Then it is possible to build the minimum WDFA recognizing $\mathcal{L}$ in $O(UB \cdot \sigma \cdot m \log m)$ time.*

Proof. Let $\{\alpha_1, ..., \alpha_m\}$ be a fingerprint of $\mathcal{L}$ and let $\mathcal{D}_{\mathcal{L}}$ be the minimum DFA recognizing $\mathcal{L}$. We can assume without loss of generality that $\alpha_1 \prec ... \prec \alpha_m$. We build the minimum Wheeler automaton $\mathcal{D}^W_{\mathcal{L}} = (Q, \alpha_1, \delta, F, \Sigma)$, where the set of states is $Q = \{\alpha_1, ..., \alpha_m\}$ and the set of final states is $F = \{\alpha_j : \alpha_j \in \mathcal{L}\}$. The transition function δ can be computed as follows. For all $1 \leq j \leq m$ and for all $c \in \Sigma$, check whether $\alpha_j \cdot c \in \text{Pref}(\mathcal{L})$. If $\alpha_j \cdot c \notin \text{Pref}(\mathcal{L})$, there are no edges labeled c that exit from α_j. If instead $\alpha_j \cdot c \in \text{Pref}(\mathcal{L})$, in order to define $\delta(\alpha_j, c)$ we just have to determine the $\equiv^c_{\mathcal{L}}$-class of the string $\alpha_j \cdot c$ (see Theorem 9). We first locate the position of $\alpha_j \cdot c$ in the intervals defined by $\alpha_1 \prec ... \prec \alpha_m$ using a binary search. There are three possible cases.

1. $\alpha_j \cdot c \preceq \alpha_1$. Then by the properties of $\equiv^c_{\mathcal{L}}$ it easily follows $\alpha_j \cdot c \equiv^c_{\mathcal{L}} \alpha_1$ and we define $\delta(\alpha_j, c) = \alpha_1$.

2. $\alpha_m \preceq \alpha_j \cdot c$. Similarly to the previous case, we have $\alpha_j \cdot c \equiv^c_{\mathcal{L}} \alpha_m$ and we define $\delta(\alpha_j, c) = \alpha_m$.

3. There exists s such that $\alpha_s \preceq \alpha_j \cdot c \preceq \alpha_{s+1}$. It can not be the case that both $\alpha_j c \not\equiv_{\mathcal{L}} \alpha_s$ and $\alpha_j c \not\equiv_{\mathcal{L}} \alpha_{s+1}$, since $\{\alpha_1, ..., \alpha_m\}$ is a fingerprint of $\mathcal{L}$ and $\equiv^c_{\mathcal{L}}$-classes are convex sets in $\text{Pref}(\mathcal{L})$. Hence we distinguish three cases.

   (a) $\alpha_s \equiv_{\mathcal{L}} \alpha_j \cdot c \not\equiv_{\mathcal{L}} \alpha_{s+1}$. Then $\alpha_j \cdot c \equiv^c_{\mathcal{L}} \alpha_s$ and we define $\delta(\alpha_j, c) = \alpha_s$.

   (b) $\alpha_s \not\equiv_{\mathcal{L}} \alpha_j \cdot c \equiv_{\mathcal{L}} \alpha_{s+1}$. Then $\delta(\alpha_j, c) = \alpha_{s+1}$.

(c) $\alpha_s \equiv_{\mathcal{L}} \alpha_j \cdot c \equiv_{\mathcal{L}} \alpha_{s+1}$. Since $\{\alpha_1, ..., \alpha_m\}$ is a fingerprint of $\mathcal{L}$, we know that strings between α_s and α_{s+1} must belong to either the $\equiv^c_{\mathcal{L}}$ class of α_s or to the $\equiv^c_{\mathcal{L}}$ class of α_{s+1}. Since $\alpha_s \equiv_{\mathcal{L}} \alpha_{s+1}$, it is either $c = \text{end}(\alpha_j c) = \text{end}(\alpha_s)$, in which case $\alpha_j \cdot c \equiv^c_{\mathcal{L}} \alpha_s$ and we define $\delta(\alpha_j, c) = \alpha_s$, or $c = \text{end}(\alpha_{s+1})$, in which case $\alpha_j \cdot c \equiv^c_{\mathcal{L}} \alpha_{s+1}$ and we define $\delta(\alpha_j, c) = \alpha_{s+1}$ (where by $\text{end}(\beta)$ we denote the last letter of the string β, for $\beta \in \Sigma^+$).

The time complexity of the described algorithm is the following: determining whether $\alpha_j \cdot c \in \text{Pref}(\mathcal{L})$ and, if it does, locating its position among $\alpha_1 \prec ... \prec \alpha_m$ using binary search requires $O(UB + UB \cdot \log m)$ time. To determine the $\equiv_{\mathcal{L}}$-class of $\alpha_j \cdot c$ it is sufficient to check its run on $\mathcal{D}_{\mathcal{L}}$ using $O(UB)$ time, for a total time of $O(UB \cdot \log m)$. We need to perform these operations for each state $\alpha_i \in Q$ and for each character $c \in \Sigma$, amounting to a total time of $O(m \cdot \sigma \cdot UB \cdot \log m)$. □

To complete the construction of the minimum WDFA, we show how to extract a *fingerprint* of a Wheeler language $\mathcal{L}$ starting from its minimum DFA. We first need to prove the following Lemma.

Lemma 24. *Given a DFA $\mathcal{D}$ with n states, a state q and a string $\gamma \notin I_q$ with $|\gamma| \leq UB = n^2 + n$, we can find in polynomial time, if it exists, the greatest (smallest) string in I_q that is smaller (greater) than γ (w.r.t. the co-lexicographic order of the strings) and has length at most UB.*

Proof. As proved in [16, 35], we can extract in polynomial time a $n \times$UB table storing, for each $(i, j) \in n \times$ UB, the smallest and the greatest string in I_{q_i} of length at most j. Given a string α, we use the notation $\alpha^-[i]$ to denote the i-th to last character of α (or ε if $i > |\alpha|$), and the notation α^-_i to denote the suffix of α of length i. In particular we have $\alpha^-_{i+1} = \alpha^-[i+1] \cdot \alpha^-_i$. In this Lemma we are interested only in strings with length less than UB, therefore every string (subset of strings) that will be mentioned has to be intended as an element (subset, respectively) of $\Sigma^{\leq \text{UB}} = \{\alpha \in \Sigma^* : \ |\alpha| \leq \text{UB}\}$.

We want to find the greatest string in I_q that is smaller than γ. Note that if γ is the suffix of a string α, then $\gamma \prec \alpha$ so we do not have to consider strings ending with γ. Note also that the greatest string smaller than γ must maximize the length of the longest suffix it has in common with γ. Therefore, we look for all the states of $\mathcal{D}$ starting from which it is possible to read the longest proper suffix of γ that ends in q. To do that, for each $1 \leq i < |\gamma|$ we build the set $S_i = \{p \in Q : \ p \overset{\gamma_i}{\rightsquigarrow} q\}$. We start from the set $S_0 = \{q\}$ and to build S_{i+1} from S_i we simply follow backward the edges labeled $\gamma^-[i+1]$. Every time we determine a set S_i, we check if there exists at least one incoming edge with a label strictly less than $\gamma^-[i+1]$. If this is the case, we keep in memory S_i as the last set we built with such property; previously stored sets can be overwritten. This procedure ends either when we find an S_i that is empty or when we successfully build the last set $S_{|\gamma|-1}$. If we did not store any of the S_i we have built, then there is no string in I_q smaller than γ. If instead we have stored at least one S_i, we consider the last one stored (that is, the only one that has not been overwritten), say S_k. Clearly, the computation of any string in I_q smaller than γ that maximizes the length of the longest suffix it has in common with γ must reach a state of S_k at its k-th to last step. Therefore, let c be the greatest label smaller than $\gamma^-[k+1]$ that enters S_k (note that c must exists since we stored S_k), and let S be the set of states that can reach S_k by an edge labeled c. Using the table described at the very beginning of this lemma, we can easily find, if it exists, the greatest string $\bar{\alpha}$ of length at most UB$-(k+1)$ that can reach a state of S. Then, the greatest string in I_q that is smaller than γ is $\bar{\alpha} \cdot c \cdot \gamma_k$.

To find the smallest string in I_q that is greater than γ, we split the problem into two sub-problems: 1) find the smallest string in I_q that is greater than γ but does not have γ as a suffix and 2) find the smallest string in I_q that has γ as a suffix. The first problem is a symmetric version of the one discussed above, and can be solved in a similar way: we use exactly the same sets S_i, but this time we store a set S_i if there exists at least one incoming edge with a label strictly greater than $\gamma^-[i+1]$. To also solve the second problem, instead of stopping when computing $S_{|\gamma|-1}$ we carry on and compute $S_{|\gamma|}$. We do this since the following conditions hold: there exists at least one string in I_q that has γ as a suffix iff $S_{|\gamma|}$ is not empty and there is at least one string of length at most $UB - |\gamma|$ that can reach a state of $S_{|\gamma|}$. If $S_{|\gamma|} \neq \emptyset$, we use again the table to determine, if it exists, the smallest string $\bar{\beta}$ of length at most $UB - |\gamma|$ that can reach a state of $S_{|\gamma|}$. Lastly, we compare $\bar{\beta} \cdot \gamma$ with the string obtained by solving the first problem and we choose the smaller one. □

As a last step, we describe an algorithm (Algorithm 1) that generates a fingerprint of a language $\mathcal{L}$ starting from the minimum DFA $\mathcal{D}_\mathcal{L}$. The algorithm uses the subroutines described in Lemma 24: given the minimum DFA $\mathcal{D}_\mathcal{L}$ with set of states $Q = \{q_0, \ldots, q_{n-1}\}$ and two strings $m, m' \in \text{Pref}(\mathcal{L}(\mathcal{D}))$ with $m \in I_{q_k}$ (for some $0 \leq k \leq n-1$),

- MinMaxPair returns the set of pairs $(m_0, M_0), \ldots, (m_{n-1}, M_{n-1})$, where m_i is the co-lexicographically smallest string in I_{q_i} of length at most UB $= n^2 + n$, and M_i is the greatest.
- GreatestSmaller(m, m') returns the greatest string in I_{q_k} smaller than m' of length at most UB.
- SmallestGreater(m, m') returns the smallest string in I_{q_k} greater than m' of length at most UB.

Notice that m is needed in the last two subroutines in order to identify k.

Algorithm 1 Min DFA to fingerprint

Require: The minimum DFA $\mathcal{D}_\mathcal{L}$ recognizing $\mathcal{L}$
Ensure: A fingerprint of $\mathcal{L}$

$\tau \leftarrow$ MinMaxPairs ▷ We initialize a set of $|\mathcal{D}_\mathcal{L}|$ pairs of strings

while there exist $c = (m, M), c' = (m', M') \in \tau$ such that $m \prec m' \prec M$ **do**
  $M_1 \leftarrow$ GreatestSmaller(m, m')
  $m_2 \leftarrow$ SmallestGreater(m, m')
  $\tau \leftarrow \tau \setminus \{c\}$
  $\tau \leftarrow \tau \cup \{c_1, c_2\}$, where $c_1 = (m, M_1), c_2 = (m_2, M)$
end while

$\tau \leftarrow$ Expand(τ)
return the first component of each element of τ

After exiting the while cycle, the algorithm uses the subroutine Expand that, given a set of pairs τ with components in Pref$(\mathcal{L})$, does the following. For each pair $(m, M) \in \tau$ such that end$(m) \neq$ end(M), it checks the incoming edges of the state $q = \delta(q_0, m)$ of $\mathcal{D}_\mathcal{L}$. For each character c such that end$(m) \prec c \prec$ end(M), it

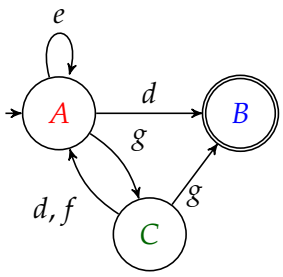


FIGURE 2.6: The minimum DFA recognizing $\mathcal{L}$.

selects a state q' such that $\delta(q', c) = q$—if any exists—and finds, using reachability, a string α_c such that $\delta(q_0, \alpha_c) = q'$. Then, it adds to τ the pair $(\alpha_c \cdot c, \alpha_c \cdot c)$. As a last step, it replaces the pair (m, M) with the pairs (m, m) and (M, M). Note that if end(m) =end(M), then Expand leaves the pair (m, M) unchanged. Figure 2.7 illustrates—on a high level—how the algorithm works, showing the evolution of the set τ for the language $\mathcal{L}$ with 3 different Myhill-Nerode classes A, B, C—colored in red, blue and green respectively—whose minimum automaton is depicted in Figure 2.6.

Each line represents the set Pref$(\mathcal{L})$ co-lexicographically ordered. The Wheeler classes of $\mathcal{L}$ are $W_1, \ldots, W_7$; the classes W_2, W_3, W_4 are distinct due to the ending character of the strings they contain: strings in W_2 end with d, strings in W_3 end with e and strings in W_4 end with f. For the sake of readability, we will use the improved upper bound UB$= 2n$ instead of UB$= n^2 + n$, hence considering strings of length at most 6.

τ_0: at the beginning we have 3 pairs—one for each Myhill-Nerode class. These pairs, output by the routine MinMaxPairs, are the following: $(m_A, M_A) = (\varepsilon, gfgfgf)$, $(m_B, M_B) = (d, gfgfgg)$, $(m_C, M_C) = (g, egfgfg)$. In Figure 2.7, the two components of each pair are linked.

τ_1: at the first iteration of the while cycle, the algorithm spots two "intersecting" pairs $(m_A, M_A), (m_B, M_B)$ and splits the former into $(m_A, M_{A1}), (m_{A2}, M_A)$. The string M_{A1} is the greatest string smaller than d that reaches A in $\mathcal{D}_\mathcal{L}$ of length at most 6; that is, $M_{A1} = \varepsilon$. The string m_{A2} is the smallest string greater than d that reaches A in $\mathcal{D}_\mathcal{L}$ of length at most 6; that is, $m_{A2} = gd$.

τ_2: at the second iteration, the algorithm spots two intersecting pairs (m_B, M_B), (m_{A2}, M_A) and splits the former into (m_B, M_{B1}),(m_{B2}, M_B). The string M_{B1} is the greatest string smaller than gd that reaches B in $\mathcal{D}_\mathcal{L}$ of length at most 6; that is, $M_{B1} = egfgfd$. The string m_{B2} is the smallest string greater than gd that reaches B in $\mathcal{D}_\mathcal{L}$ of length at most 6; that is, $m_{B2} = gg$.

Expand(τ_2): in τ_2 all pairs are ordered—that is, they do not intersect. The subroutine Expand replaces the pair $(m_{A2}, M_A) \in \tau_2$ with the pairs $(m_{A2}, m_{A2}), (e, e), (M_A, M_A)$, where e is the shortest string that ends with e and reaches A in $\mathcal{D}_\mathcal{L}$.

The first component of each pair of Expand(τ_2) is our output fingerprint:

$$\varepsilon,\ d,\ gd,\ e,\ gfgfgf,\ g,\ gg.$$

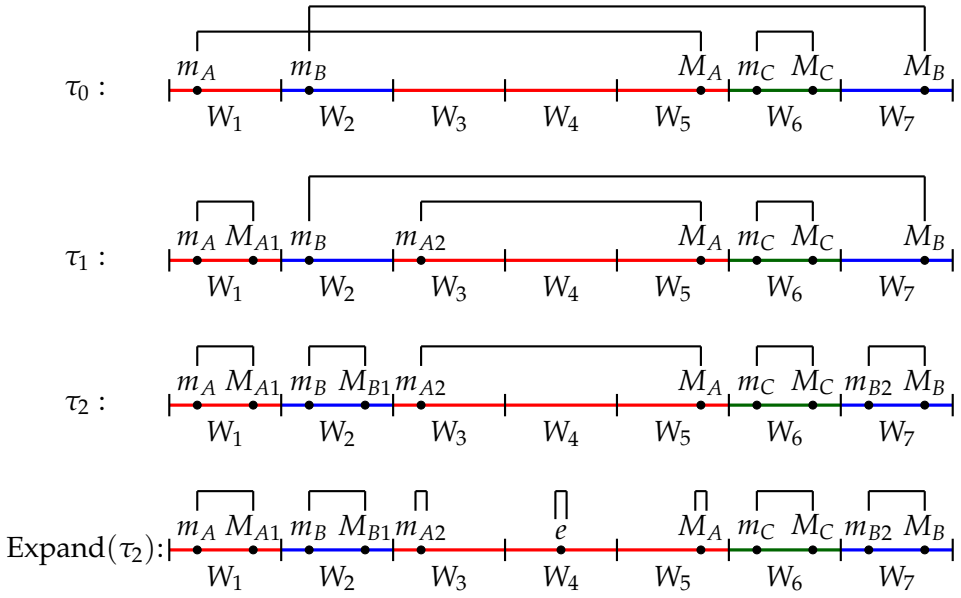


FIGURE 2.7: The evolution of the set τ for the language $\mathcal{L}$ with 3 different Myhill-Nerode classes A, B, C—colored in red, blue and green respectively.

To prove the correctness of the algorithm, we show some properties that hold for the pairs in τ. Let τ_i denote the set τ at the end of the i-th iteration of the while cycle, let $c = (m, M), c' = (m', M')$ be the pairs selected on Line 2 and let $c_1 = (m_{c_1}, M_{c_1}) = (m, M_1), c_2 = (m_{c_2}, M_{c_2}) = (m_2, M)$ be the pairs added to τ_i on Line 6. That is, $m \prec m' \prec M$ and the pairs c_1, c_2 will substitute the pair c at the end of the while cycle. A trivial property that can be proved by induction on i is the following: if $d = (m_d, M_d) \in \tau_i$, then $m_d \preceq M_d$ and $m_d \equiv_{\mathcal{L}} M_d$. Moreover, by construction we have $M_{c_1} \equiv_{\mathcal{L}} m_{c_2} \equiv_{\mathcal{L}} m$. The next property is an invariant of the while cycle, where we use the following definition.

Definition 13. Let $d = (m_d, M_d)$ and $f = (m_f, M_f)$ be two pairs such that $m_d \preceq M_d$ and $m_f \preceq M_f$. We say that d and f are *ordered* if either $M_d \prec m_f$ or $M_f \prec m_d$.

Proposition 25. *Let $d = (m_d, M_d)$ and $f = (m_f, M_f)$ be two distinct pairs in τ_i such that $m_d \equiv_{\mathcal{L}} m_f$. Then d and f are ordered.*

Proof. We proceed by induction on i. Base step: if $i = 0$ there is nothing to prove, since in τ_0 distinct pairs have components belonging to different $\equiv_{\mathcal{L}}$-classes. Induction step: assume that the invariant holds for τ_i and let $d = (m_d, M_d), f = (m_f, M_f) \in \tau_{i+1}$ be two pairs such that $m_d \equiv_{\mathcal{L}} m_f$. Let $c = (m, M), c' = (m', M')$ be the pairs selected on Line 2 at stage i and let $c_1 = (m_{c_1}, M_{c_1}) = (m, M_1), c_2 = (m_{c_2}, M_{c_2}) = (m_2, M)$ be the pairs added to τ_i on Line 6.
If $\{d, f\} \cap \{c_1, c_2\} = \emptyset$, then $d, f \in \tau_i$ and the thesis follows from the inductive hypothesis.
If $\{d, f\} = \{c_1, c_2\}$, then d, f are ordered by construction.
If $\{d, f\} \cap \{c_1, c_2\} \neq \emptyset$ and $\{d, f\} \neq \{c_1, c_2\}$, suppose w.l.o.g. that $d \in \{c_1, c_2\}$ and $f \notin \{c_1, c_2\}$. From $m_f \equiv_{\mathcal{L}} m_d \in \{m_{c_1}, m_{c_2}\}$ and $m_{c_1} = m \equiv_{\mathcal{L}} m_{c_2}$ it follows that $m_f \equiv_{\mathcal{L}} m$. Since $c, f \in \tau_i$, $m_f \equiv_{\mathcal{L}} m$, and $c = (m, M)$, from the inductive hypothesis it follows that c, f are ordered, thus it immediately follows that both c_1, f and c_2, f —and hence d, f— are ordered since the intervals $[m_{c_1}, M_{c_1}], [m_{c_2}, M_{c_2}]$ are subsets of $[m, M]$. □

In particular, for a fixed i, if two pairs $(m, M), (m', M') \in \tau_i$ have components belonging to the same Myhill-Nerode equivalent class, then it can not be the case that $m \prec m' \prec M$. Therefore, it holds that the two pairs $c = (m, M), c' = (m', M')$ selected on Line 2 are such that $m \not\equiv_{\mathcal{L}} m'$.

Definition 14. Let W be a Wheeler class of a Wheeler language $\mathcal{L}$ and let τ be a set of pairs. We say that W is *represented* by τ iff there exists a pair $d = (m_d, M_d) \in \tau$ such that either $m_d \in W$ or $M_d \in W$. We say that W is *enclosed* by τ iff there exists a pair $d = (m_d, M_d) \in \tau$ such that $W \subseteq [m_d]_{\equiv_{\mathcal{L}}}$ and $m_d \prec W \prec M_d$.

Note that a Wheeler class can either be represented or enclosed by a pair, but not both.

Proposition 26. *Let W be a Wheeler class of a Wheeler language $\mathcal{L}$. Then, for all $i \geq 0$, W is either represented or enclosed by τ_i.*

Proof. We proceed by induction on i. For $i = 0$, consider the Myhill-Nerode class C of $\mathcal{L}$ containing W. From Remark 14 we have that the Wheeler classes are ordered, hence let W_m, W_M be the smallest and greatest, respectively, Wheeler classes included in C. From Lemma 22 it follows that there exist strings $\alpha_m \in W_m$ and $\alpha_M \in W_M$ of length at most $n^2 + n$. Therefore, the pair (m_C, M_C) created by the routine MinMaxPair is such that $m_C \in W_m$ and $M_C \in W_M$. Hence, either $W \in \{W_m, W_M\}$ so W is represented, or $W_m \prec W \prec W_M$ so W is enclosed.

Now assume the thesis holds for τ_i and let W be a Wheeler class. If W is represented by τ_{i+1}, there is nothing to prove. If instead W is not represented by τ_{i+1}, note that each component of a pair in τ_i is still a component of some pair in τ_{i+1}. Therefore W can not be represented by τ_i either, and from the inductive hypothesis it follows that W is enclosed by τ_i. Let $d = (m_d, M_d) \in \tau_i$ be a pair that encloses W, that is, $W \subseteq [m_d]_{\equiv_{\mathcal{L}}}$ and $m_d \prec W \prec M_d$. Let $c = (m, M), c' = (m', M')$ be the pairs selected on Line 2, with c substituted by $c_1 = (m_1, M_1), c_2 = (m_2, M_2)$ in line 6. If $d \neq c$ then $d \in \tau_{i+1}$, thus W is enclosed by τ_{i+1}. If $d = c$, then $m_d = m$ and $m \prec W \subseteq [m]_{\equiv_{\mathcal{L}}}$. Since c, c' are not ordered, from Proposition 25 it follows that $m \not\equiv_{\mathcal{L}} m'$ and, since W is convex in $\mathrm{Pref}(\mathcal{L})$ (see Remark 11), there are two possible cases. If $W \prec m'$, then either W is the greatest Wheeler class smaller than m', in which case W is represented by M_1—the second component of c_1—; or W is a Wheeler class such that $m \prec W \prec M_1$, in which case W is enclosed by c_1. Similarly, if $m' \prec W$ then either W is represented by m_2 or W is enclosed by c_2. In all cases, W is either represented or enclosed by τ_{i+1} and the thesis follows. □

Definition 15. Let $\mathcal{L}$ be a Wheeler language and let $d = (m_d, M_d)$ be a pair of strings, with $m_d, M_d \in \mathrm{Pref}(\mathcal{L})$. We denote by $w(d)$ the number of Wheeler classes W of $\mathcal{L}$ such that $m_d \prec W \prec M_d$.
Given a set of pairs τ, we denote by $w(\tau)$ the value

$$w(\tau) := \sum_{c \in \tau} w(c).$$

Note that, for any pair d, the value $w(d)$ is a non negative integer. Moreover, if $w(d) = 0$ then m_d, M_d belong to the same Wheeler class.

Proposition 27 (Termination of Algorithm 1)**.** *If $\mathcal{L}$ is Wheeler, on input $\mathcal{D}_{\mathcal{L}}$ Algorithm 1 terminates.*

Proof. To prove that the algorithm terminates, we show that at each iteration of the while cycle the value $w(\tau)$ strictly decreases—that is, $w(\tau_{i+1}) < w(\tau_i)$ for all i. The

only pairs that are not common to τ_i and τ_{i+1} are c, c_1, c_2, where c, c_1, c_2 are the pairs mentioned in line 2 and 6 of algorithm 1; thus it is enough to prove that $w(c_1) + w(c_2) < w(c)$. Note that if W is a Wheeler class such that $m \prec W \prec M$, it can not be the case that both $m \prec W \prec M_1$ and $m_2 \prec W \prec M$ holds, since by construction we have $M_1 \prec m_2$. Therefore we have $w(c_1) + w(c_2) \leq w(c)$. To prove that the inequality is strict, consider the Wheeler class containing m', say W' (where $c' = (m', M')$ is the pair mentioned in line 2 of the algorithm 1). From $m \equiv_{\mathcal{L}} M$ and $m \not\equiv_{\mathcal{L}} m'$ (see Proposition 25) it follows that $m \prec W' \prec M$, thus W' contributes to the value $w(c)$. From $M_1 \prec m' \prec m_2$ it follows that neither $m \prec W' \prec M_1$ nor $m_2 \prec W' \prec M$ holds. Therefore, W' does not contribute to either $w(c_1)$ nor $w(c_2)$, so we have $w(c_1) + w(c_2) \leq w(c) - 1$. □

Lemma 28. *Let $d = (m_d, M_d) \in \tau_i$ be a pair of strings and let W be a Wheeler class such that $m_d \prec W \prec M_d$ but $W \nsubseteq [m_d]_{\equiv_{\mathcal{L}}}$. Then there exist a pair $d_i = (m_i, M_i) \in \tau_i$ such that d, d_i meet the while condition of Line 2 of Algorithm 1.*

Proof. Consider a Wheeler class W such that $m_d \prec W \prec M_d$ and $W \nsubseteq [m_d]_{\equiv_{\mathcal{L}}}$, for some pair $d = (m_d, M_d) \in \tau_i$. We claim that we can find a pair $f = (m_f, M_f) \in \tau_i$ such that $m_d \prec m_f \prec M_d$ or $m_f \prec m_d \prec M_f$, that is d, f meet the while condition of Line 2.

Proposition 26 states that W is either represented or enclosed by τ_i. If W is represented by τ_i, let $f = (m_f, M_f) \in \tau_i$ be a pair that represents W, that is, either $m_f \in W$ or $M_f \in W$. If $m_f \in W$ it immediately follows that $m_d \prec m_f \prec M_d$. If instead $M_f \in W$ we have $m_d \prec M_f \prec M_d$, with $m_d \not\equiv_{\mathcal{L}} m_f$. If $m_d \prec m_f$ we have $m_d \prec m_f \preceq M_f \prec M_d$, otherwise we have $m_f \prec m_d \prec M_f$. In all cases, d, f meet the while condition of Line 2.

If W is enclosed by τ_i, let $f = (m_f, M_f) \in \tau_i$ be a pair that encloses W, that is, $m_f \prec W \prec M_f$ with $W \subseteq [m_f]_{\equiv_{\mathcal{L}}}$. From $m_d \prec W \prec M_d$ and $m_f \prec W \prec M_f$ it immediately follows that $m_d \prec M_f$ and $m_f \prec M_d$, thus, since $m_d \not\equiv_{\mathcal{L}} m_f$ and hence $m_d \neq m_f$, either $m_d \prec m_f \prec M_d$ or $m_f \prec m_d \prec M_f$ holds. In both cases, d, f meet the while condition of Line 2. □

An immediate consequence of Lemma 28 is the following corollary.

Corollary 28.1. *Let τ_p be the last set built before exiting the while cycle of Algorithm 1 and let $d = (m_d, M_d) \in \tau_p$ be a pair of strings. Then, the only Wheeler classes W such that $m_d \prec W \prec M_d$ are the enclosed ones—if any exists.*

Proposition 29 (Correctness of Algorithm 1)**.** *If $\mathcal{L}$ is Wheeler, on input $\mathcal{D}_{\mathcal{L}}$ Algorithm 1 outputs a fingerprint of $\mathcal{L}$.*

Proof. Let τ_p be the last set built before exiting the while cycle of Algorithm 1. We need to prove that the collection of the first components of the pairs in Expand(τ_p) is a fingerprint of $\mathcal{L}$, that is Expand(τ_p) represents each Wheeler class of $\mathcal{L}$ exactly once. Proposition 26 states that every Wheeler class is either represented or enclosed by τ_p, and clearly classes represented by τ_p are also represented by Expand(τ_p). Moreover, it can be easily proved, by induction on i, that given a Wheeler class W of $\mathcal{L}$, there exists at most one pair in τ_i that represents W. Hence each Wheeler classes represented by τ_p is represented by exactly one pair in τ_p. It remains to be proved that each Wheeler class enclosed by τ_p is represented by Expand(τ_p).

Clearly, all pairs in τ_p are ordered, otherwise we would still be inside the while cycle.

Consider now a pair $d = (m_d, M_d) \in \tau_p$ that encloses a Wheeler class W. Since all pairs in τ_p are ordered, $m_d \equiv_{\mathcal{L}} M_d$ holds, and, from Corollary 28.1, every

Wheeler class W between m_d and M_d is a subset of $[m_d]_{\equiv_{\mathcal{L}}}$, it follows that every string $\alpha \in \text{Pref}(\mathcal{L})$ that belongs to the interval $[m_d, M_d]$ also belongs to the Myhill-Nerode class $[m_d]_{\equiv_{\mathcal{L}}}$: otherwise, the class $[\alpha]_{\equiv^c_{\mathcal{L}}} \subseteq [m_d, M_d]$ would contradict Corollary 28.1. Therefore, the only possibility for W to be a Wheeler class that differs from $[m_d]_{\equiv^c_{\mathcal{L}}}$ is that the strings in W end with a character different from $\text{end}(m_d)$. The Expand routine checks, for each pair in τ_p, the existence of such Wheeler classes by looking at the incoming edges of the state $\delta(q_0, m_d)$ and proceeds to select one representative for each.

Since $\text{Expand}(\tau_p)$ represents every Wheeler class and the two components of any pair in $\text{Expand}(\tau_p)$ belong to the same Wheeler class, we can safely extract the first components of the pairs in $\text{Expand}(\tau_p)$ to obtain a fingerprint of $\mathcal{L}$. □

Notice that in Proposition 23 a generic fingerprint of the Wheeler language $\mathcal{L}$ is needed to compute $\mathcal{D}^W_{\mathcal{L}}$. Algorithm 1 does indeed produce a fingerprint, but it actually does something more: for every Wheeler class of $\mathcal{L}$, it outputs its smaller and its greatest string among the ones of length at most $\text{UB} = n^2 + n$. As a matter of fact, this is the intuition that led to Algorithm 1: since Wheeler classes are convex sets of strings, only two strings are needed to represents each class —the smallest and the greatest. This way, we are able to represent an entire Wheeler class using only a finite number of strings, which enable us to process sets of strings in one go. Lemma 22 is of utmost importance for this result: it states that we can consider only strings of length at most UB, which ensures the existence of a smaller and a greatest string with the required property.

Notice also that we can make a simple comparison between Hopcroft's algorithm for computing the minimum DFA (see [36]) and the proposed algorithm for computing the minimum WDFA. Hopcroft's algorithm starts with a large set of states and at each step collapses those it realizes to be equivalent. Algorithm 1 does the exact opposite: it starts with a small set of states (those of the minimum automaton) and divides them whenever it realizes they violate the convexity condition. To obtain the minimum WDFA, Algorithm 1 must therefore ensure to make the minimum number of cuts when deciding to divide states.

2.3 Generalized Wheelerness

Changing the underlying order of the alphabet might turn a Wheeler language into a not Wheeler one and vice versa. For instance, consider again the Wheeler languages $\mathcal{L}_d$ and the regular (but not Wheeler) language $\mathcal{L}_b$ depicted in Figure 1.5. If we change the order of Σ from $a \prec c \prec d \prec f$ to $a \prec d \prec c \prec f$, the Wheeler language $\mathcal{L}_d$ turns into a non-Wheeler language (isomorphic to $\mathcal{L}_b$ under the isomorphism φ between alphabets that fixes characters a, c, f and sends d into $\varphi(d) = b$). Hence, by not fixing an a priori order of the alphabet Σ we extend the class of languages.

Definition 16 (Generalized Wheelerness)**.** A NFA $\mathcal{A}$ over the alphabet Σ is called a Generalized Wheeler Automaton (GWNFA) if and only if there exists an ordering of the elements of Σ that makes $\mathcal{A}$ Wheeler.
A language $\mathcal{L}$ is called *generalized Wheeler* (for short GW) if and only if there exists a GWNFA that recognizes $\mathcal{L}$.

How big is the class of GW languages? Clearly, it extends the class of Wheeler languages which is a subclass of star-free languages (see Proposition 18). Therefore

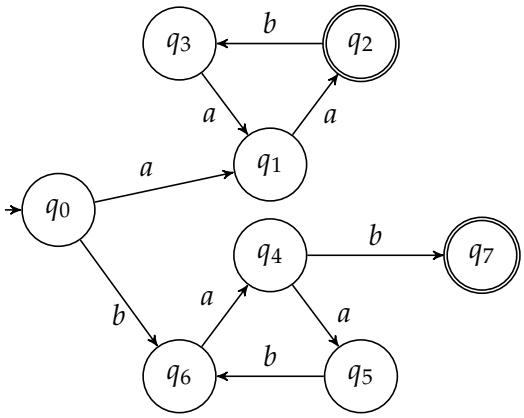


FIGURE 2.8: The minimum DFA recognizing $\mathcal{L} = a(aba)^*a \cup ba(aba)^*b$. The absence of counters implies that $\mathcal{L}$ is star-free.

we are interested in discovering the relationship between the classes of GW and star-free languages. In the next proposition, we show that the former is a proper subclass of the latter, hence GW languages must be studied separately.

Proposition 30. *If* $|\Sigma| \geq 2$, *then the set* $\{\mathcal{L} \subseteq \Sigma^* : \mathcal{L}$ *is GW*$\}$ *is a proper subset of* $\{\mathcal{L} \subseteq \Sigma^* : \mathcal{L}$ *is star-free*$\}$.

Proof. First we prove that GW languages are a subclass of star-free languages. From proposition 18 it follows that Wheeler languages are a subclass of star-free languages. Since star-free expressions, and thus star-free languages, are closed under permutations of the alphabet, even GW languages must be a subclass of star-free languages.

To prove that the inclusion is strict, we show an example of a star-free language that is not GW. Let a, b be two distinct characters of Σ, and consider the language $\mathcal{L} = a(aba)^*a \cup ba(aba)^*b$. As shown in Figure 2.8, $\mathcal{L}$ is star-free. But $\mathcal{L}$ is not GW: consider the sequence $(\alpha_i)_{i\geq 2}$ with $\alpha_{2n} = a(aba)^n$ and $\alpha_{2n+1} = ba(aba)^n$. Since, for all i, the string α_i is a prefix of α_{i+1}, independently from how a and b are ordered we have

$$aaba \prec baaba \prec aabaaba \prec baabaaba \prec \cdots \prec a(aba)^i \prec ba(aba)^i \prec \ldots$$

Moreover, for all i we have $a(aba)^i \not\equiv_{\mathcal{L}} ba(aba)^i$, since $a(aba)^i \cdot a$ belongs to $\mathcal{L}$ but $ba(aba)^i \cdot a$ does not. We can then apply Lemma 15 to conclude that $\mathcal{L}$ is not Wheeler. Since this result does not depend on the order of the alphabet, $\mathcal{L}$ is not GW. □

As stated in Lemma 7, we can decide in polynomial time whether a DFA is Wheeler. On the contrary, deciding whether a NFA is Wheeler is NP-complete (see [17]). Since deciding whether an NFA is a GWNFA adds a level of complexity to deciding whether an NFA is Wheeler, it is reasonable to expect that the former problem is at least as difficult as the latter. We will show that in fact both problems have the same degree of complexity (NP-complete), as stated in Proposition 31. We actually prove a stronger result in Proposition 32: even deciding whether a DFA is a GWNFA is NP-complete. Since the proof of Proposition 32 is complicated, we decided to present also its weaker version, i.e. Proposition 31, which uses a simpler reduction from the problem of deciding whether a NFA is Wheeler. It is worth noticing that the proof of Proposition 31 can be adapted to work even on DFAs, hence

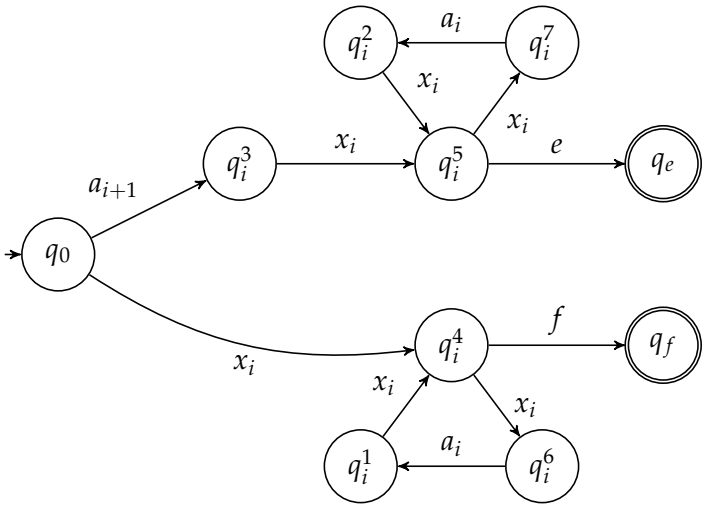


FIGURE 2.9: The gadget G_i, connected to q_0 and to the sinks q_e and q_f.

giving an alternative way to prove that deciding whether a DFA is a GWNFA is NP-complete. Nonetheless, Proposition 32 is still stronger, since it also proves that deciding whether a DFA recognizes a GW language is NP-complete.

Proposition 31 (GWNFA complexity)**.** *Let $\mathcal{A}$ be a NFA. Deciding whether $\mathcal{A}$ is a GWNFA is NP-complete.*

Proof. The problem is in NP, since we can use non-determinism to guess both the order of the alphabet and the Wheeler order among the states and then check whether such orders makes the NFA Wheeler in polynomial time. This can be done by checking whether this two orders satisfy conditions (W1) and (W2) in Definition 6.

To prove the hardness, we show a polynomial reduction from the problem of deciding whether a NFA is Wheeler. Let $\mathcal{A}$ be a NFA with initial state q_0, over the alphabet $\Sigma = \{a_1 \dots, a_\sigma\}$ ordered by the relation $a_1 \prec \cdots \prec a_\sigma$. We want to build a new automaton $\mathcal{A}'$ such that $\mathcal{A}'$ is a GWNFA if and only if $\mathcal{A}$ is Wheeler. $\mathcal{A}'$ will be an automaton of size $|\mathcal{A}| + O(\sigma)$ over the alphabet Σ' of size $O(\sigma)$. Notice that, since Wheelerness implies input-consistency, we may suppose w.l.o.g. that $\mathcal{A}$ is input consistent.

The automaton $\mathcal{A}'$ will be built starting from $\mathcal{A}$ and adding extra states and transitions. We define the new alphabet as

$$\Sigma' = \{a_1, \dots, a_\sigma, x_1, \dots, x_{\sigma-1}, e, f\},$$

with $x_i, e, f \notin \Sigma$, and we add two final states q_e and q_f. We then build $\sigma - 1$ gadgets, one for each $a_i \in \Sigma$ with $i \neq \sigma$, each one connected to $\mathcal{A} \cup \{q_e, q_f\}$ as depicted in Figure 2.9. This completes the construction of the automaton $\mathcal{A}'$. Notice that $\mathcal{A}'$ is input-consistent but is not deterministic (even if $\mathcal{A}$ is) due to the new Σ-transitions starting from q_0.

We want to show that $\mathcal{A}$ is Wheeler according to the order $(\Sigma, \prec)$ if and only if $\mathcal{A}'$ is a GWNFA.
($\Longrightarrow$) Define the order $\prec'$ over Σ' by setting

$$a_1 \prec' \dots \prec' a_\sigma \prec' x_1 \prec' \dots \prec' x_{\sigma-1} \prec' e \prec' f.$$

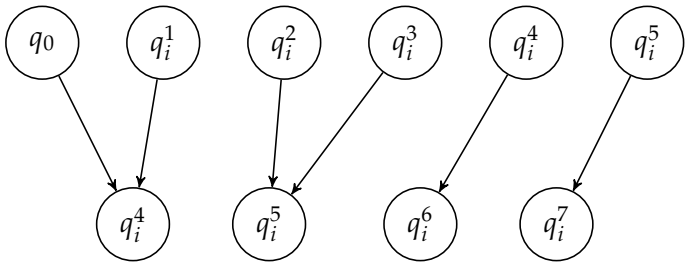


FIGURE 2.10: All the edges labeled x_i of $\mathcal{A}'$. Starting states in the top layer are ordered from left to right, as well as the ending states in the bottom layer.

We show that $\mathcal{A}'$ is Wheeler according to $(\Sigma', \prec')$ by ordering its states. Since $\mathcal{A}$ is Wheeler, there already exists an order of its states that makes $\mathcal{A}$ Wheeler. Therefore, we simply need to extend this order to the states of $\mathcal{A}'$. First of all, we will order, for each i, the states with incoming edge x_i. Since $x_i \notin \Sigma$, the only states to compare are the one belonging to the gadget G_i, which we order according to their superscript:

$$q_i^4 < q_i^5 < q_i^6 < q_i^7.$$

Secondly, for each $1 \leq i \leq \sigma$ we will sort the states with incoming edges labeled a_i. Notice that the automaton $\mathcal{A}$ must contain states —at least one— with incoming label a_i so we need to consider such states as well. For each state q of $\mathcal{A}$ with incoming edges labeled a_i we set

$$q_{i-1}^3 < q < q_i^1 < q_i^2;$$

notice that for $i = 1$ the state q_{i-1}^3 does not exist, and for $i = \sigma$ states q_i^1, q_i^2 do not exist: in this cases, we erase the missing states from the inequality above. Lastly, whenever there are two states whose incoming edges have distinct labels we order them according to such labels, with q_0 preceding every other state.

The extended ordered that we defined is total and satisfies by construction condition (W1) of Definition 6. To prove that it also satisfies condition (W2), we consider, for each character $a \in \Sigma'$, the set of edges of $\mathcal{A}'$ labeled a.
If $a \in \{e, f\}$ there is nothing to prove: there is only one state with incoming edges labeled e (f), namely q_e (q_f), hence condition (W2) is trivially satisfied.
If $a = x_i$ for some i, the only edges labeled x_i are the one belonging to gadget G_i. These edges are depicted in Figure 2.10 along with their starting and ending states. In the figure, starting states are arranged in the top layer whereas ending states (possibly duplicated if they were both starting and ending ones) are arranged in the bottom one, with all edges going from the top to the bottom. Since states in both layers are ordered from left to right (according to the extended order we defined), condition (W2) —applied only to this set of edges— is satisfied if and only if edges never cross, which is indeed true.
If $a = a_i$ for some i, from the assumption that $\mathcal{A}$ is Wheeler it follows that pairs of edges of $\mathcal{A}$ labeled a_i satisfy condition (W2). Therefore it is sufficient to select any edge (q, a_i, q') of $\mathcal{A}$ and show that it satisfies condition (W2) along with edges labeled a_i introduced in gadgets G_{i-1} and G_i. These edges are depicted in Figure 2.11 and they never cross, hence condition (W2) holds. Notice that this is still true when $q = q_0$, that is, when the edge (q, a_i, q') starts from the initial state.

The order of the states of $\mathcal{A}'$ that we described makes $\mathcal{A}'$ Wheeler with respect to

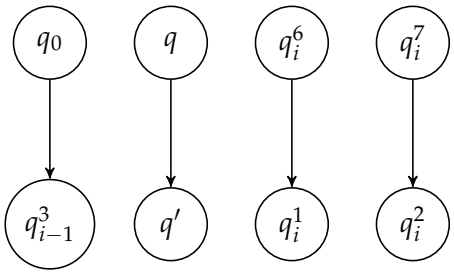


FIGURE 2.11: The edges labeled a_i of $\mathcal{A}'$ which were not edges of $\mathcal{A}$, along with the edge (q, a_i, q') of $\mathcal{A}$. Starting states in the top layer are ordered from left to right, as well as the ending states in the bottom layer. If $i = 1$ we need to erase state q_{i-1}^3 and hence also state q_0 and the edge connecting them. If $i = \sigma$ we need to erase states q_i^1, q_i^2 and hence also states q_i^6, q_i^7 and the edges connecting them.

$(\Sigma', \prec')$, hence making it a GWNFA.
($\Longleftarrow$) If $\mathcal{A}'$ is a GWNFA, then there exists an order $\prec'$ over Σ' that makes $\mathcal{A}'$ Wheeler. Since $\mathcal{A}$ is a sub-automaton of $\mathcal{A}'$, it follows that even $\mathcal{A}$ is Wheeler according to $\prec'$. Let $\widetilde{\prec}$ be the restriction of $\prec'$ over the alphabet Σ; we want to show that $\widetilde{\prec}$ is the same order as $\prec$. Assume by contradiction that $\widetilde{\prec} \neq \prec$. If for all $1 \leq i < \sigma$ we have $a_i \widetilde{\prec} a_{i+1}$, then $\widetilde{\prec} = \prec$, a contradiction. Hence there exists $1 \leq i < \sigma$ such that $a_{i+1} \widetilde{\prec} a_i$. Since $\prec'$ extends $\widetilde{\prec}$, this implies that $a_{i+1} \prec' a_i$. We will show that $\mathcal{A}'$ is not Wheeler according to $\prec'$, a contradiction. Define the strings $\mu := x_i$, $\nu := a_{i+1}x_i$ and $\gamma := x_i a_i x_i$. From $\mu \dashv \gamma$ and $a_{i+1} \prec' a_i$ we have $\mu, \nu \prec \gamma$. The string γ labels two cycles in $\mathcal{A}'$ starting from two distinct states, i.e. q_i^4 and q_i^5. Moreover, μ and ν label two paths that start from the initial state q_0 and end in q_i^4 and q_i^5 respectively. Since q_i^4 and q_i^5 are not Myhill-Nerode equivalent, we can apply Theorem 16 to conclude that $\mathcal{L}(\mathcal{A}')$ is not Wheeler according to $\prec'$. Hence $\mathcal{A}'$ cannot be Wheeler with respect to $\prec'$, a contradiction. Thus $\widetilde{\prec}$ and $\prec$ coincide.
We have shown that $\mathcal{A}$ is Wheeler according to $\prec'$, and that $\prec'$ extends $\prec$. Therefore we can conclude that $\prec' = \prec$, hence $\mathcal{A}$ is Wheeler according to $\prec$. □

To prove the next proposition, we will show a reduction to the Betweenness problem (see Definition 8).

Proposition 32 (GWDFA and GW languages hardness)**.** *Let $\mathcal{L} \subseteq \Sigma^*$ be a language and $\mathcal{A}$ be a DFA. Both the problems of deciding whether $\mathcal{A}$ is a GWNFA and deciding whether $\mathcal{L}$ is GW are NP-complete.*

Proof. We can prove that both problems are in NP using an argument similar to the one employed in the proof of Proposition 31: we simply guess an order of the states in $\mathcal{A}$ (in the minimum automaton for $\mathcal{L}(\mathcal{A})$, respectively) and then use Lemma 7 (Theorem 17, respectively).

To prove the hardness, we show a polynomial reduction from the betweenness problem to both of the problems described; we will use exactly the same reduction for both problems. We start from an instance $I = (Y, K)$ of the betweenness problem, where Y is the set $Y = \{y_1 \ldots, y_n\}$ and $K \subseteq \mathcal{P}(T^3)$ is a collection of k triples $(a_1, b_1, c_1), \ldots, (a_k, b_k, c_k)$, each composed of distinct elements, for some $1 < k < n^3$. We build a DFA $\mathcal{A}$ of size $O(n + k)$, over an alphabet of size $O(n + k)$. The alphabet is $\Sigma = Y \cup \{x_1 \ldots, x_k, e, f\}$, where we introduce a new character x_i for each triple $(a_i, b_i, c_i) \in K$ and two extra "ending" characters e and f. To build $\mathcal{A}$, we start with

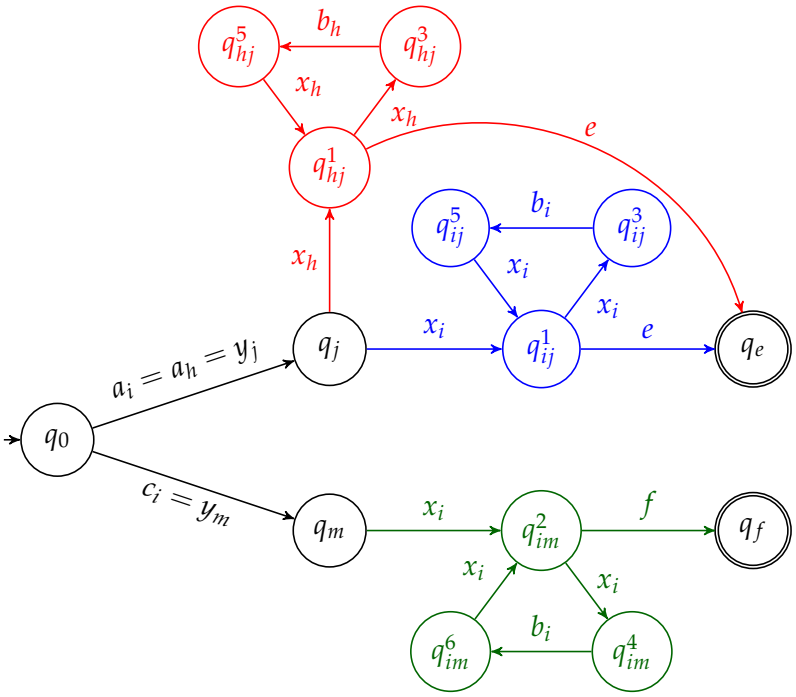


FIGURE 2.12: The gadgets G_i (in green) and G'_i (in blue) related to the i-th triple. Notice that if $a_h = a_i = y_j$ for $h \neq i$ we also have a G_h gadget (in red) different from G_i and the same holds for c_i-gadgets

the initial state q_0 connected with n states $q_1, \ldots, q_n$ through the edges (q_0, y_j, q_j) for each $1 \leq j \leq n$. We also add two sinks q_e and q_f, the only final states.

For each $i \in \{1, \ldots, k\}$, if $j \in \{1, \ldots, n\}$ is such that $a_i = y_j$ we add a gadget G_i with states $q^1_{ij}, q^3_{ij}, q^5_{ij}$ (see Figure 2.12), transitions

$$\delta(q_j, x_i) = q^1_{ij}, \quad \delta(q^1_{ij}, x_i) = q^3_{ij}, \quad \delta(q^3_{ij}, b_i) = q^5_{ij}, \quad \delta(q^5_{ij}, x_i) = q^1_{ij},$$

and $\delta(q^1_{ij}, e) = q_e$.
We repeat the same process with c_i: given the integer m such that $c_i = y_m$, we add a gadget G'_i with states $q^2_{im}, q^4_{im}, q^6_{im}$ and transitions

$$\delta(q_m, x_i) = q^2_{im}, \quad \delta(q^2_{im}, x_i) = q^4_{im}, \quad \delta(q^4_{im}, b_i) = q^6_{im}, \quad \delta(q^6_{im}, x_i) = q^2_{im}$$

and $\delta(q^2_{im}, f) = q_f$. Lastly, we remove the states among $q_1, \ldots, q_n$ that don't have outgoing edges. More formally, we define the sets $A := \{a_1, \ldots, a_k\}$ and $C := \{c_1, \ldots, c_k\}$ and we remove from $\mathcal{A}$ all the states q_j such that $y_j \notin A \cup C$. This end the construction of the automaton $\mathcal{A}$, which is an input consistent DFA.[2]

We show that the instance $I = (Y, K)$ of the betweennes problem is satisfiable if and only if $\mathcal{A}$ is a GWNFA, if and only if $\mathcal{L}(\mathcal{A})$ is GW.
($\Longrightarrow$) Since $I = (Y, K)$ is satisfiable, there exists an ordering of the elements of Y satisfying I which we present as the bijection $\pi : Y \to \{1, ..., n\}$. We order Σ as follows:

$$\pi^{-1}(1) \prec ... \prec \pi^{-1}(n) \prec x_1 \prec \cdots \prec x_k \prec e \prec f.$$

[2]Notice that if $c_h = y_j$ then $a_h \neq y_j$ because the element in the triple (a_h, b_h, c_h) are distinct. Hence, there is at most one x_i-transition leaving q_j.

This ordering induce a partial order on the states of $\mathcal{A}$ defines as follows. Following Proposition 6, to obtain a Wheeler order $<$ on $\mathcal{A}$ we need only to ensure that if q, q' are such that $q < q'$ then $I_q \prec I_{q'}$. Therefore, states with different incoming labels are ordered by such labels and we only need to order the states of $\mathcal{A}$ with the same incoming label.
For each $1 \leq i \leq k$, the only states of $\mathcal{A}$ with incoming label x_i are $q^1_{ij}, q^3_{ij}, q^2_{im}, q^4_{im}$, where j and m are integers such that $a_i = y_j$ and $c_i = y_m$. Since, by construction, the order π satisfies the instance I, then only two cases can occur: either $\pi(a_i) < \pi(b_i) < \pi(c_i)$, or $\pi(c_i) < \pi(b_i) < \pi(a_i)$. In the first case, we set $q^1_{ij} < q^2_{im} < q^3_{ij} < q^4_{im}$. To realize that this is in fact the correct order of the states, consider the following languages:

$$
\begin{aligned}
I_1 := I_{q^1_{ij}} &= \{\alpha \in \Sigma^* : \ \delta(q_0, \alpha) = q^1_{ij}\} = a_i x_i (x_i b_i x_i)^* \\
I_2 := I_{q^2_{im}} &= c_i x_i (x_i b_i x_i)^* \\
I_3 := I_{q^3_{ij}} &= a_i x_i x_i (b_i x_i x_i)^* \\
I_4 := I_{q^4_{im}} &= c_i x_i x_i (b_i x_i x_i)^*.
\end{aligned}
$$

Since, by construction, we have $a_i, b_i, c_i \prec x_i$, it follows that $I_1, I_2 \prec I_3, I_4$. Moreover, from $\pi(a_i) < \pi(b_i) < \pi(c_i)$ we also have that $I_1 \prec I_2$ and $I_3 \prec I_4$, which completes the ordering. Symmetrically, if $\pi(c_i) < \pi(b_i) < \pi(a_i)$ then we set $q^2_{im} < q^1_{ij} < q^4_{im} < q^3_{ij}$.
We still need to order the states of $\mathcal{A}$ whose incoming labels belong to Y. For each $1 \leq p \leq n$, the states with incoming label y_p belong to the sets $V^p_5 := \{q^5_{ij} : \ b_i = y_p\}$ or $V^p_6 := \{q^6_{\ell m} : \ b_\ell = y_p\}$ or V_p, where $V_p = \{q_p\}$ if q_p is a state of $\mathcal{A}$ (i.e. if $y_p \in A \cup C$) and $V_p = \emptyset$ otherwise. If $V_p = \{q_p\}$, we set q_p as the smallest state. We then sort the states of V^p_5 and V^p_6 by their first subscript, that is: $q^5_{ij} < q^6_{\ell m}$ if $i < \ell$. When the first subscript is equal, i.e. the two states that we want to confront are q^5_{ij} and q^6_{im} (with $a_i = y_j$ and $c_i = y_m$), then we set $q^5_{ij} < q^6_{im}$ if $\pi(a_i) < \pi(b_i) < \pi(c_i)$ and we set $q^6_{im} < q^5_{ij}$ if $\pi(c_i) < \pi(b_i) < \pi(a_i)$. We can check that this order is correct by considering the following incoming languages:

$$
\begin{aligned}
I_5 := I_{q^5_{ij}} &= a_i x_i x_i b_i (x_i x_i b_i)^* \\
I_6 := I_{q^6_{\ell m}} &= c_\ell x_\ell x_\ell b_\ell (x_\ell x_\ell b_\ell)^*.
\end{aligned}
$$

First, notice that if $q^5_{ij} \in V^p_5$ and $q^6_{\ell m} \in V^p_6$ then $b_i = b_\ell = y_p$. If $i < \ell$ then, by construction, we have $x_i \prec x_\ell$, hence $I_5 \prec I_6$. Symmetrically, if $\ell < i$ then we have $I_6 \prec I_5$. Lastly, if $i = \ell$ then the order between I_5 and I_6 is determined solely by a_i and $c_\ell = c_i$.

Since there is only one state with incoming label e and only one state with incoming label f, we have finished. The order described makes $\mathcal{A}$ Wheeler, thus $\mathcal{A}$ is a GWNFA and $\mathcal{L}(\mathcal{A})$ is GW.
($\Longleftarrow$) Assume that the instance $I = (Y, K)$ of the betweenness problem is unsatisfiable. We prove that $\mathcal{L}(\mathcal{A})$ is not GW. Assume by contradiction that $\mathcal{L}(\mathcal{A})$ is GW, then there exists an ordering π' of the elements of Σ such that $\mathcal{L}(\mathcal{A})$ is Wheeler according to this order. Recall that $Y \subseteq \Sigma$ and consider the order $\pi := \pi'|_Y$. Since (Y, K) is unsatisfiable, π must violate one of the constraints, i.e. there exists an $1 \leq i \leq k$

such that either $\pi(a_i), \pi(c_i) < \pi(b_i)$ or $\pi(b_i) < \pi(a_i), \pi(c_i)$. Define the strings $\mu := a_i x_i$, $\nu := c_i x_i$ and $\gamma := x_i b_i x_i$; then it is either $\mu, \nu \prec \gamma$ or $\gamma \prec \mu, \nu$ (here the co-lexicographic order $\prec$ is calculated with respect to π'). By construction, γ labels two cycles in $\mathcal{A}$ starting from two distinct states, q^1_{ij} and q^2_{im}, which are not Myhill-Nerode equivalent. Moreover, μ and ν label two paths that start from the initial state q_0 and end in q^1_{ij} and q^2_{im} respectively. We can then apply the Theorem 16 to conclude that $\mathcal{L}(\mathcal{A})$ is not Wheeler according to π', a contradiction. Therefore $\mathcal{L}(\mathcal{A})$ is not GW, which automatically implies that $\mathcal{A}$ is not a GWNFA. □

2.4 Complexity on *reduced* NFAs

Among the two possible ways of presenting regular languages by automata, that is DFAs or NFAs, computational problems tend to be significantly harder in general when referred to the non-deterministic class. Typical examples are: checking emptiness, computing the intersection, checking universality and much more. In the realm of Wheeler automata and languages a new class emerges: the class of reduced automata, formally defined below.

Definition 17. An NFA $\mathcal{A} = (Q, S, \delta, F, \Sigma)$ is called *reduced* if $q \neq p$ implies $I_q \neq I_p$.

Clearly, the class of reduced NFAs contains properly the class of DFAs. When Wheelerness is concerned, the class of reduced NFAs is interesting because it has been proved that deciding whether an NFA is Wheeler is an NP-complete problem [17], whereas deciding whether a *reduced* NFA is Wheeler turns out to be in P [16] as it is for DFAs [30]. An intuition behind this result is as follows: Proposition 5 provides a necessary condition for two states of an NFA to be ordered according to a Wheeler order, namely that their input languages are ordered with respect to the relation $\preceq_{ps}$. In a non-reduced automaton, multiple states may share the same input language. In this case, Proposition 5 is of no help in searching for a Wheeler ordering, and it becomes necessary to find, for each set of states with the same input language, an order among them that is compatible with the ordering of the remaining states. In contrast, starting from a reduced NFA eliminates this problem and we can rely on the input languages of its states to find a possible Wheeler ordering.

Clearly, any NFA can be turned into a reduced one simply by merging all the states that recognize the same input language. Finding states to be merged is complex: the language-equivalence problem for NFAs can easily be proved as complex as deciding whether two states of an NFA recognize the same input language and, therefore, the latter is PSPACE-complete.

A natural question is now whether switching from NFAs to reduced NFAs simplifies some otherwise difficult problem. In this section we prove that this is not always the case: some problems remain hard even when restricted to the class of reduced NFAs. We will prove three main results. First, the universality problem remains hard for reduced NFAs (see Lemma 33). However, since the class of reduced NFAs emerges in connection with the Wheelerness property, we may hope that hard problem related to Wheelerness become easier when restricted from NFAs to reduced NFAs. Unfortunately, we shall prove that this is not the case. Indeed, for our second result we will consider an interesting aspect of the relationship between DFAs, NFAs, and reduced NFAs related to Wheelerness and indexability. Given an NFA $\mathcal{A}$, the partial order $<_{\mathcal{A}}$, defined on the set of $\mathcal{A}$-states by

$$q <_{\mathcal{A}} q' \Leftrightarrow I_q \prec_{ps} I_{q'} \wedge I_q \neq I_{q'},$$

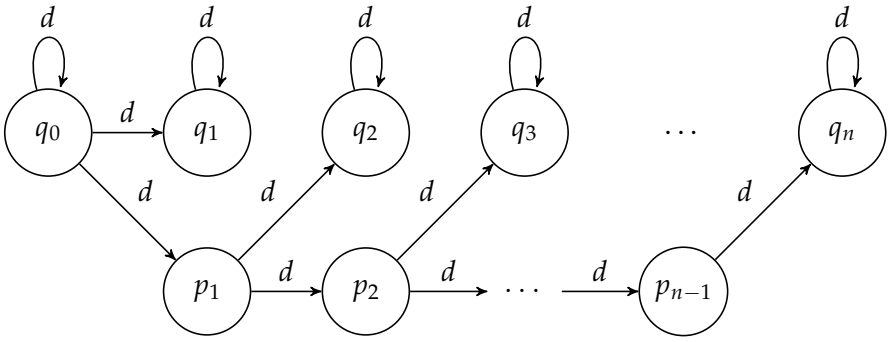


FIGURE 2.13: The automaton $\mathcal{A}'$ with only d-transitions depicted.

allows to represent $\mathcal{A}$ using an index, that is, a succint structure that supports fast matching queries [23]. The order $<_{\mathcal{A}}$ is related to Wheelerness since, for DFAs, $<_{\mathcal{D}}$ is total iff $\mathcal{D}$ is Wheeler, and, remarkably, the partial order $<_{\mathcal{D}}$ can be computed in polynomial time [23]. Here we prove that these good properties do not hold for reduced NFAs (see Theorem 34 and Figure 2.14). Third, moving from automata to languages, we will prove that deciding whether a NFA recognizes a Wheeler language is PSPACE-complete, even for reduced NFAs (see Theorem 36).

2.4.1 Universality and the ps-order for reduced NFAs

We start from the universality problem, which remains difficult even for reduced NFAs.

Lemma 33. *The universality problem for reduced NFAs is PSPACE-complete.*

Proof. This problem belongs to PSPACE, since it is a restriction of the universality problem over generic NFAs. To prove hardness, we show a reduction from the universality problem on NFAs.

Given an NFA $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$, we can assume w.l.o.g. that the initial state has no incoming edges and that it is the only state with such property. We build a new automaton $\mathcal{A}' = (Q \cup P, q_0, \delta', F, \Sigma \cup \{d\})$, where $P = \{p_1, \ldots, p_{n-1}\}$ is a set of $n-1$ new states and d is a new character. For each $q \in Q$ we add the self loop (q, d, q). If we add only these transitions, it holds that $\mathcal{L}(\mathcal{A}) = \Sigma^*$ iff $\mathcal{L}(\mathcal{A}') = (\Sigma \cup \{d\})^*$. We can now add to the automaton as many d-transitions as we please without violating the property $\mathcal{L}(\mathcal{A}) = \Sigma^*$ iff $\mathcal{L}(\mathcal{A}') = (\Sigma \cup \{d\})^*$: the right-to-left implication still holds if we only add d-transitions, whereas the left-to-right implication holds since adding transitions may only expand the recognized language, but $(\Sigma \cup \{d\})^*$ is already maximal (with respect to the inclusion). Therefore we add the transitions (q_0, d, q_1) and (q_0, d, p_1). Moreover, for each $1 \leq i \leq n-1$ we add the transitions (p_i, d, q_{i+1}) and (p_i, d, p_{i+1}) (see Figure 2.13).

To conclude the proof that the reduction is correct, we need to show that $\mathcal{A}'$ is reduced. Since q_0 has no incoming edges except from (q_0, d, q_0), we have

$$I_{q_0} = d^*$$
$$I_{p_i} = d^* \cdot d^i \quad \text{for } 1 \leq i \leq n-1.$$

Since $\mathcal{A}$ was trimmed and since each $q \in Q \setminus \{q_0\}$ is not an initial state, we have $I_q \cap \Sigma^+ \neq \emptyset$ for each $q \in Q \setminus \{q_0\}$. Thus $I_q \neq I_p$ for each $q \in Q \setminus \{q_0\}$ and for

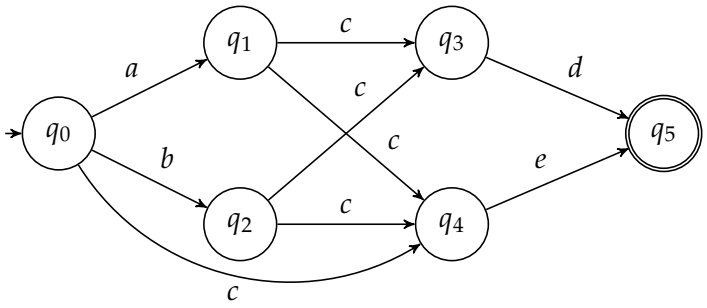


FIGURE 2.14: A reduced NFA $\mathcal{A}$ such that the order $<_{\mathcal{A}}$ is total. The only non-trivial inequality to verify is $q_3 <_{\mathcal{A}} q_4$, which follows from the fact that $I_{q_3} = \{ac, bc\} \prec_{ps} \{ac, bc, c\} = I_{q_4}$. Notice that $\mathcal{A}$ is not Wheeler: states q_3, q_4 can not be ordered. In fact, assume that $q_3 < q_4$ holds in the Wheeler order. Then, we could apply W2 to the edges $(q_2, c, q_3), (q_1, c, q_4)$ to derive that $q_2 < q_1$, which contradicts W1. A similar contradiction derives if we assume $q_4 < q_3$ and we consider the edges $(q_1, c, q_3), (q_2, c, q_4)$.

each $p \in P \cup \{q_0\}$. Moreover, for each $1 \leq i < j \leq n$ we have $d^i \in I_{q_i} \setminus I_{q_j}$, hence $I_{q_i} \neq I_{q_j}$. □

We will use the previous lemma to prove our second result. The partial order $<_{\mathcal{A}}$ is defined using the family of input languages $\{I_q : q \in Q\}$ and the prefix-suffix order $\preceq_{ps}$ (see Definition 7), which can be lifted to the collection of states of an NFA.

Definition 18. Given two states q and p of an NFA $\mathcal{A}$, we say that $q <_{\mathcal{A}} p$ iff $I_q \preceq_{ps} I_p$ and $I_q \neq I_p$.

Note that if $\mathcal{D}$ is a DFA, then $<_{\mathcal{D}}$ simplifies:

$$q <_{\mathcal{D}} p \iff \forall \alpha \in I_q \, \forall \beta \in I_p (\alpha \prec \beta),$$

and this order satisfies the properties of a Wheeler order, with the exception of not necessarily being total. As a matter of fact, it can be proved that the DFA $\mathcal{D}$ is Wheeler if and only if $<_{\mathcal{D}}$ is a total order. In general, for a reduced NFA $\mathcal{A}$, this equivalence does not hold, since there are examples of reduced, non-Wheeler NFAs where $<_{\mathcal{A}}$ is total (see Figure 2.14).

Nevertheless, knowing the $<_{\mathcal{A}}$ order of an NFA allows us to build a compressible index, so a natural question is how difficult is to compute such order. In the following theorem we show that this is not an easy task.

Theorem 34. *Given two states q and p of an NFA $\mathcal{A}$, deciding whether $q <_{\mathcal{A}} p$ is PSPACE-complete. The same result holds even if $\mathcal{A}$ is reduced.*

Proof. First of all we need to prove that the problem is in PSPACE. We will show instead that its complement is in NPSPACE, then the thesis follows from Savitch's Theorem [37], which states that NPSPACE = PSPACE, and the fact that PSPACE is closed under complementation.

The complement of our problem consist of answering to the question whether $q \not<_{\mathcal{A}} p$. To do so, first we check whether $I_q = I_p$. As we have already mentioned,

this problem is in PSPACE, so we can get the answer in polynomial space. If $I_q = I_p$, then $q \not<_{\mathcal{A}} p$ and we answer "yes". Otherwise, we have

$$q <_{\mathcal{A}} p \iff \forall \alpha \in I_q \, \forall \beta \in I_p \big(\{\alpha, \beta\} \not\subseteq I_q \cap I_p \Rightarrow \alpha \prec \beta \big),$$

or equivalently

$$q \not<_{\mathcal{A}} p \iff \exists \alpha \in I_q \, \exists \beta \in I_p \big(\{\alpha, \beta\} \not\subseteq I_q \cap I_p \wedge \beta \prec \alpha \big).$$

Let d be the number of states of the DFA $\mathcal{D}$ generated by the determinization of $\mathcal{A}$ using the powerset construction; clearly it holds $d \leq 2^n$, where $n = |\mathcal{A}|$. We claim that if $q \not<_{\mathcal{A}} p$, then there exist two strings α, β of length at most $d^2 + d$ such that

$$\alpha \in I_q \ \wedge \ \beta \in I_p \ \wedge \ \{\alpha, \beta\} \not\subseteq I_q \cap I_p \ \wedge \ \beta \prec \alpha. \tag{2.4}$$

Assume that α, β satisfy (2.4), with either $|\alpha|$ or $|\beta|$ (possibly both) greater than $d^2 + d$. We distinguish two cases.
1) The last d^2 characters of α and β differs; this also includes the case where the length of one them is strictly less than d^2. Suppose that $|\beta| \geq |\alpha|$ (if $|\beta| < |\alpha|$ just switch the role of α and β in what follows). Consider the $d + 1$ states of $\mathcal{D}$ visited by reading the first d characters of β. Since $\mathcal{D}$ has d states, at least one of them appears twice, implying that we visited a cycle. By erasing from the first d characters of β the factor corresponding to such cycle, we obtain a string β' reaching the same state as β in $\mathcal{D}$. Since $\mathcal{D}$ is obtained from $\mathcal{A}$ using the powerset construction, this means that β and β' reach the same set of states in $\mathcal{A}$. Hence, α and β' still satisfy (2.4).
2) The last d^2 characters of α and β coincide; in particular $|\alpha|, |\beta| \geq d^2$. Consider the last $d^2 + 1$ states $r_0, ..., r_{d^2}$ of $\mathcal{D}$ visited by reading the string α, and the last $d^2 + 1$ states $p_0, ..., p_{d^2}$ visited by reading the string β. Since $\mathcal{D}$ has only d states, there must exist $0 \leq i, j \leq d^2$ with $i < j$ such that $(r_i, p_i) = (r_j, p_j)$, implying that α and β visited two cycles labeled by the same string. By erasing from the last d^2 characters of α and β the factor corresponding to such cycles, the same argument used in case 1) shows that we obtain two strings α', β' which still satisfy (2.4).
In both cases, we were able to shorten the length of the longest string. By repeating this process as many times as needed, we will eventually obtain two strings both shorter than $d^2 + d$, with $d \leq 2^n$.

Now that we have bounded the length of α, β with the constant $2^{2n} + 2^n$, we can use non-determinism to guess, bit by bit, the length of α and β and store this guessed information in two counters a, b respectively, using $O\big(\log(2^{2n} + 2^n)\big) = O(n)$ space for each. These counters determine which string among α and β is longer and we start guessing the characters of such longest string from the left to the right, decreasing by one its counter whenever we guess a character. Note that we are not storing the guessed characters, since this would require too much space. When the counter reaches the same value of the other counter, we start guessing the characters of both the first and the second string at the same time and we carry on until both counters reach the value 0. While guessing the characters of α (respectively, β) we update at each step the set of states of $\mathcal{A}$ reachable from q_0 by reading the currently guessed prefix of α (β), so that in the end we obtain the sets $\delta(q_0, \alpha)$ and $\delta(q_0, \beta)$. With this information, we can check whether $\alpha \in I_q$ and $\beta \in I_p$ and $\{\alpha, \beta\} \not\subseteq I_q \cap I_p$. To complete checking condition (2.4), we need to show how to decide whether $\beta \prec \alpha$.

To confront co-lexicographically α and β, we use a variable ρ that indicates whether α is less, equal or greater than β. We initialize ρ based on the counters

a, b as follows:

$$\rho := \begin{cases} = & \text{if } a = b \\ \dashv & \text{if } a < b \\ \vdash & \text{if } b < a. \end{cases}$$

We leave ρ unchanged until we start guessing simultaneously the characters of α and β. When we guess the character c_1 for α and the character c_2 for β, we set

$$\rho := \begin{cases} \prec & \text{if } c_1 \prec c_2 \\ \succ & \text{if } c_1 \succ c_2 \\ \rho & \text{if } c_1 = c_2. \end{cases}$$

Note that if at the end ρ has value $\dashv$, it means that $\alpha \dashv \beta$, thus $\alpha \prec \beta$. Similarly, if ρ has value $\vdash$ then $\beta \prec \alpha$. Otherwise, we have $\alpha \,\rho\, \beta$. Thus we are always able to determine the co-lexicographic order of α and β. Therefore, deciding whether $q \not\leftarrow_{\mathcal{A}} p$ is a problem in PSPACE, and so it is its complement.

To prove hardness, we show a reduction from the universality problem over (reduced) NFAs.

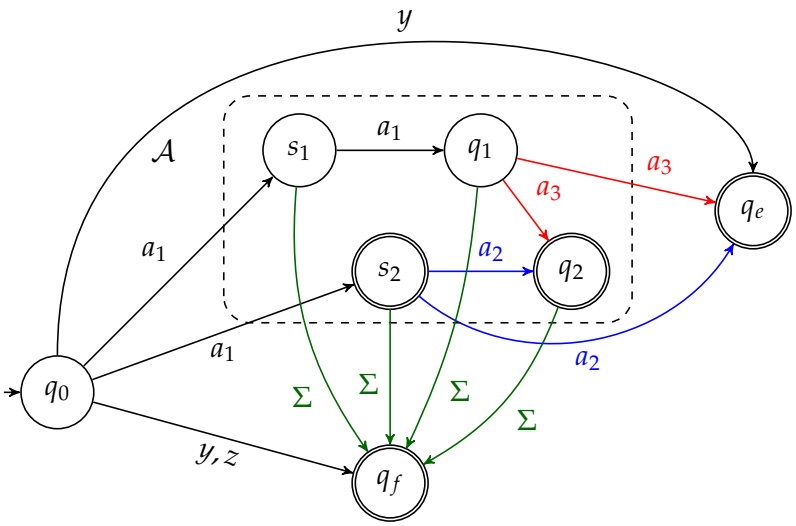


FIGURE 2.15: The automaton $\mathcal{A}'$ built starting from the automaton $\mathcal{A}$ with $S = \{s_1, s_2\}$ recognizing the language $\mathcal{L} = \{\varepsilon, a_2, a_1a_3\}$. Edges entering a final state in $\mathcal{A}$ have been duplicated and redirected to q_e. Green edges are labeled $\Sigma = \{a_1, a_2, a_3\}$.

Given an NFA $\mathcal{A} = (Q, S, \delta, F, \Sigma)$ recognizing the language $\mathcal{L} = \mathcal{L}(\mathcal{A})$, where $Q = \{q_1, \ldots, q_n\}$, S is the set of initial states, and $\Sigma = \{a_1, \ldots, a_\sigma\}$, we build a new NFA $\mathcal{A}' = (Q', q_0, \delta', F \cup \{q_e, q_f\}, \Sigma')$ by adding a new initial state q_0 and two final states $\{q_e, q_f\}$ (see Figure 2.15). The new alphabet is $\Sigma' = \Sigma \cup \{y, z\}$, where $a_j \prec y \prec z$ for each $1 \leq j \leq \sigma$. For each $q_i \in S$, we add a transition from q_0 to q_i labeled a_1. Adding q_0 has the sole purpose of having an initial state without incoming edges. Note that we can not make the usual assumption that $\mathcal{A}$ has only one initial state which moreover has no incoming edges: if we start from a reduced NFA and we build an equivalent NFA with the required property, there is no guarantee that the

new automaton will still be reduced. The state q_e represents the new final state that gathers all the strings in $a_1 \cdot (\mathcal{L} \setminus \{\varepsilon\})$. To achieve this goal, for each transition $(q_i, a_j, q_{i'})$ of δ such that $q_{i'} \in F$ we add a new transition (q_i, a_j, q_e). The state q_f gathers all the strings in $a_1 \cdot \mathrm{Pref}(\mathcal{L}) \cdot \Sigma$, and this can be easily achieved by adding a transition (q_i, a_j, q_f) for each $i \geq 1$ and $j \geq 1$. Lastly, we add the transitions (q_0, y, q_e), (q_0, y, q_f) and (q_0, z, q_f). In this way, if $\mathcal{A}$ is reduced then $\mathcal{A}'$ is also reduced, because $I_{q_0} = \{\varepsilon\}$, for each $i \geq 1$ it holds $I^{\mathcal{A}'}_{q_i} = a_1 \cdot I^{\mathcal{A}}_{q_i}$, the states q_e, q_f are the only ones that can read the string y, and q_f is the only state that can read the string z.

Let $\mathcal{L}_\varepsilon$ denote the language $\mathcal{L} \setminus \{\varepsilon\}$. By construction, we have

$$I_{q_e} = a_1 \cdot \mathcal{L}_\varepsilon \cup \{y\}$$
$$I_{q_f} = a_1 \cdot \mathrm{Pref}(\mathcal{L}) \cdot \Sigma \cup \{y\} \cup \{z\}.$$

We want to show that $\mathcal{L} = \Sigma^*$ iff $q_e <_{\mathcal{A}'} q_f \wedge \Sigma \cup \{\varepsilon\} \subseteq \mathcal{L}$.

Note that $\Sigma \cup \{\varepsilon\} \subseteq \mathcal{L}$ is a necessary condition for $\mathcal{L}$ to be universal, and such condition can be checked in polynomial time using reachability on $\mathcal{A}$, therefore the reduction is still polynomial.

($\Rightarrow$) If $\mathcal{L} = \Sigma^*$, it clearly follows that $\Sigma \cup \{\varepsilon\} \subseteq \mathcal{L}$. Moreover we have $\mathrm{Pref}(\mathcal{L}) \cdot \Sigma = \Sigma^+$ and we obtain

$$I_{q_e} = a_1 \cdot \Sigma^+ \cup \{y\}$$
$$I_{q_f} = a_1 \cdot \Sigma^+ \cup \{y\} \cup \{z\}.$$

It follows immediately that $q_e <_{\mathcal{A}'} q_f$.

($\Leftarrow$) Assume that $q_e <_{\mathcal{A}'} q_f \wedge \Sigma \cup \{\varepsilon\} \subseteq \mathcal{L}$. Note that—actually, in general— $\mathcal{L}_\varepsilon \subseteq \mathrm{Pref}(\mathcal{L}) \cdot \Sigma$ holds. We first prove that from the hypothesis it follows $\mathcal{L}_\varepsilon = \mathrm{Pref}(\mathcal{L}) \cdot \Sigma$. Assume by contradiction that $\mathcal{L}_\varepsilon \neq \mathrm{Pref}(\mathcal{L}) \cdot \Sigma$ and let β be a string in $\mathrm{Pref}(\mathcal{L}) \cdot \Sigma \setminus \mathcal{L}_\varepsilon$. Then we have

$$y \in I_{q_e}, \quad a_1 \cdot \beta \in I_{q_f}, \quad \{y, a_1 \cdot \beta\} \nsubseteq I_{q_e} \cap I_{q_f}$$

but $y \succ a_1 \cdot \beta$, a contradiction. Thus $\mathcal{L}_\varepsilon = \mathrm{Pref}(\mathcal{L}) \cdot \Sigma$.

We can then prove by induction on $|\alpha|$ that $\alpha \in \Sigma^+$ implies $\alpha \in \mathcal{L}_\varepsilon$. If $|\alpha| = 1$ then $\alpha \in \Sigma$ and by hypothesis we have $\Sigma \subseteq \mathcal{L}_\varepsilon$. If $|\alpha| = n + 1 > 1$, then $\alpha = \alpha' \cdot a_j$ for some $\alpha' \in \Sigma^+$ and some $a_j \in \Sigma$. By induction hypothesis we have $\alpha' \in \mathcal{L}_\varepsilon \subseteq \mathrm{Pref}(\mathcal{L})$, and from $\mathcal{L}_\varepsilon = \mathrm{Pref}(\mathcal{L}) \cdot \Sigma$ it follows $\alpha \in \mathcal{L}_\varepsilon$. Therefore, we have $\Sigma^+ \subseteq \mathcal{L}_\varepsilon$. From the assumption $\Sigma \cup \{\varepsilon\} \subseteq \mathcal{L}$ it follows that $\varepsilon \in \mathcal{L}$, hence $\Sigma^* \subseteq \mathcal{L}$ and the claim follows.

This concludes the reduction from the universality problem to our problem over general NFAs. Since the construction described preserves the reduced-ness of the starting automaton, it also works as a reduction from the universality problem over reduced NFAs to our problem over reduced NFAs. In Lemma 33 we proved that the former problem is PSPACE-complete, thus proving that the latter is also PSPACE-complete. □

An immediate corollary of this theorem is the following.

Corollary 34.1. *Deciding whether an NFA is reduced is PSPACE-complete.*

Proof. To prove that the problem is in PSPACE, note that $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$ is reduced iff, for all $q, p \in Q$, $q \neq p$ implies $I_q \neq I_p$. Therefore, it is sufficient to check $O(n^2)$ times whether $I_q = I_p$, where $n = |Q|$. As we have already mentioned, the problem of deciding whether $I_q = I_p$ belongs to PSPACE, thus the thesis follows.

To prove hardness, we combine the reductions shown in Lemma 33 and Theorem 34: given a NFA $\mathcal{A}$, we build a NFA $\mathcal{A}''$ such that $\mathcal{L}(\mathcal{A}) = \Sigma^*$ iff $\mathcal{A}''$ is *not* reduced and $\Sigma_{+d} \cup \{\varepsilon\} \subseteq \mathcal{L}'$, where $\Sigma_{+d} = \Sigma \cup \{d\}$.
We first apply the reduction shown in Lemma 33 to build a *reduced* automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \Sigma^*$ iff $\mathcal{L}(\mathcal{A}') = \Sigma^*_{+d}$. We set $\mathcal{L}' := \mathcal{L}(\mathcal{A}')$. Then, we apply the reduction showed in Theorem 34 to the automaton $\mathcal{A}'$, but we remove the edge (q_0, z, q_f); we call this new automaton $\mathcal{A}''$. The languages recognized by q_e and q_f change as follows:

$$I_{q_e} = a_1 \cdot \mathcal{L}'_\varepsilon \cup \{y\}$$
$$I_{q_f} = a_1 \cdot \mathrm{Pref}(\mathcal{L}') \cdot \Sigma \cup \{y\}.$$

Since $\mathcal{A}'$ is a reduced automaton and the states q_e and q_f are the only ones with an incoming edge labeled y, it immediately follows that $\mathcal{A}''$ is *not* reduced iff $I_{q_e} = I_{q_f}$. Applying the same argument we used in Theorem 34, we can conclude that $\mathcal{L}' = \Sigma^*_{+d}$ iff $I_{q_e} = I_{q_f} \wedge \Sigma_{+d} \cup \{\varepsilon\} \subseteq \mathcal{L}'$. Summarizing, we have that $\mathcal{L}(\mathcal{A}) = \Sigma^*$ iff $\mathcal{A}''$ is *not* reduced and $\Sigma_{+d} \cup \{\varepsilon\} \subseteq \mathcal{L}'$. Since we can check the last condition in polynomial time, the reduction is complete and the thesis follows since PSPACE is closed under complementation. □

2.4.2 Recognizing language-Wheelerness starting form a reduced NFA

Finally, we will prove our last result of this section: deciding whether a NFA recognizes a Wheeler language is PSPACE-complete, in contrast with the deterministic case, even if the NFA is reduced.

In order to prove this result we will need a technical lemma which will make use of a generalization of Theorem 16, whose proof can be adapted to work on generic DFAs —not just the minimum. Since we need this more general result but its proof would be both long and technical, we will instead prove the following lemma, which is easier to prove although the bound given in condition 4 is not optimal. This is not a big loss, since we will only use the fact that this bound is polynomial in n.

Lemma 35. *Let $\mathcal{D} = (Q, q_0, \delta, F, \Sigma)$ be a DFA recognizing the language $\mathcal{L}$, with $n = |\mathcal{D}|$. The following equivalence holds:*
$\mathcal{L}$ is not Wheeler if and only if there exist μ, ν and γ in Σ^*, with $\gamma \not\asymp \mu, \nu$, such that:

1. *$\mu \not\equiv_{\mathcal{L}} \nu$ and they label paths from q_0 to states u and v, respectively;*
2. *γ labels two cycles, one starting from u and one starting from v;*
3. *$\mu, \nu \prec \gamma$ or $\gamma \prec \mu, \nu$.*

The length of the strings μ, ν and γ satisfying the above conditions can be bounded:

4. *$|\mu|, |\nu| \leq |\gamma| \leq (n^3 + 2n^2 + n + 2) \cdot n^2$.*

Proof. Let $\mathcal{D}_{\mathcal{L}} = (\hat{Q}, \hat{q}_0, \hat{\delta}, \hat{F}, \Sigma)$ be the minimum DFA recognizing $\mathcal{L}$. Clearly $\mathcal{D}_{\mathcal{L}}$ has at most n states.
($\Longleftarrow$) We assume there exist strings μ, ν, γ such that conditions 1-3 hold. From condition 2 it follows that $\delta(q_0, \mu) = \delta(q_0, \mu\gamma)$, thus $\mu \equiv_{\mathcal{L}} \mu\gamma$. Therefore, in $\mathcal{D}_{\mathcal{L}}$ we also have $\hat{\delta}(\hat{q}_0, \mu) = \hat{\delta}(\hat{q}_0, \mu\gamma)$. Similarly, it holds $\hat{\delta}(\hat{q}_0, \nu) = \hat{\delta}(\hat{q}_0, \nu\gamma)$. It follows that μ, ν, γ satisfy condition 1-3 of Theorem 16, hence $\mathcal{L}$ is not Wheeler.
($\Longrightarrow$) We assume $\mathcal{L}$ is not Wheeler. Since $\mathcal{L}$ is not Wheeler, let $\hat{\mu}, \hat{\nu}, \hat{\gamma}$ be three strings satisfying conditions 1-4 of Theorem 16. The DFA $\mathcal{D}_{\mathcal{L}}$ has at most n states, hence the

length of $\hat{\mu}, \hat{\nu}$ and $\hat{\gamma}$ is bounded by $n^3 + 2n^2 + n + 2$. We have $\hat{\mu}\hat{\gamma}^* \subseteq \text{Pref}(\mathcal{L})$, so let $t_0 = q_0, t_1, \ldots, t_\ell$ be a run of $\hat{\mu}\hat{\gamma}^n$ over $\mathcal{D}$. We set $m := |\hat{\mu}|$ and $g := |\hat{\gamma}|$, and consider the list of $n+1$ states

$$t_m,\ t_{m+g},\ t_{m+2g},\ \ldots,\ t_{m+ng} = t_\ell$$

Since $\mathcal{D}$ has n states, there must exist two integers $0 \leq h < k \leq n$ such that $t_{m+hg} = t_{m+kg}$. That is, there exists a state $p := t_{m+hg}$ such that $p \in \delta\left(q_0, \hat{\mu}\hat{\gamma}^h\right)$ and $\hat{\gamma}^{k-h}$ labels a cycle starting from p. We can repeat the same argument for a run of $\hat{\nu}\hat{\gamma}^n$ over $\mathcal{D}$ to find a state r and two integers h', k' such that $r \in \delta(q_0, \hat{\nu}\hat{\gamma}^{h'})$ and $\hat{\gamma}^{k'-h'}$ labels a cycle starting from r. If we define the constant h'' as the minimum multiple of $(k-h)\cdot(k'-h')$ greater than n, we can prove that $h'' \leq n^2$: from $k-h, k'-h' < n$ it follows that $(k-h)\cdot(k'-h') < n^2$. Therefore, if $(k-h)\cdot(k'-h') \geq n$ we have $h'' = (k-h)\cdot(k'-h') < n^2$. If instead $(k-h)\cdot(k'-h') < n$ we have $h'' \leq n\cdot(k-h)\cdot(k'-h') < n^2$. By construction $\hat{\gamma}^{h''}$ labels both a cycle starting from p and one starting from r. We then define the strings

$$\mu := \hat{\mu}\hat{\gamma}^h$$
$$\nu := \hat{\nu}\hat{\gamma}^{h'}$$
$$\gamma := \hat{\gamma}^{h''},$$

which satisfy conditions 2 and 3. Note that we have chosen a h'' such that $|\gamma| \geq n > |\mu|, |\nu|$, so that $\gamma \not\prec \mu, \nu$. Condition 4 is satisfied since $|\hat{\gamma}| \leq n^3 + 2n^2 + n + 2$ and $h'' \leq n^2$. Lastly, condition 1 is satisfied since the strings $\hat{\mu}$ and $\hat{\mu}\hat{\gamma}^h$ lead to the same state of $\mathcal{D}_\mathcal{L}$, thus $\hat{\mu} \equiv_\mathcal{L} \hat{\mu}\hat{\gamma}^h$. Similarly, we have $\hat{\nu} \equiv_\mathcal{L} \hat{\nu}\hat{\gamma}^{h'}$. The thesis then follows from $\hat{\mu} \not\equiv_\mathcal{L} \hat{\nu}$. □

The polynomial bound given by condition 4 of Lemma 35 allows us to design an algorithm that decides whether a given DFA recognizes a Wheeler language: using dynamic programming it is possible to keep track of all the relevant paths and cycles inside the DFA and check, in polynomial time, whether there exists three strings satisfying the conditions of the lemma (see [30]).

Things change if, instead of a DFA, we are given an NFA: given a NFA, building an equivalent DFA and checking (language) Wheelerness on it would not be feasible due to the possible blow-up in the number of states. We show that the problem of deciding whether an NFA recognizes a Wheeler language is indeed hard, but does not necessarily require exponential space to be solved. Instead, the problem turns out to be PSPACE-complete.

Theorem 36. *Given a NFA $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$, deciding whether the language $\mathcal{L} := \mathcal{L}(\mathcal{A})$ is Wheeler is PSPACE-complete. The statement is true even if restricted to reduced NFAs.*

Proof. First of all we need to prove that the problem is in PSPACE. As we have already stated in the proof of Theorem 34, it is sufficient to show that its complement is in NPSPACE.

Let $\mathcal{D}$ be the automaton obtained by the powerset construction applied to $\mathcal{A}$ with dimension $d = |\mathcal{D}| \leq 2^n$. We prove that we can check the conditions in Lemma 35 for the automaton $\mathcal{D}$, without building it, using polynomial space. We use nondeterminism to guess the sequences of bits representing the lengths of μ, ν, and γ, and store this guessed information in three counters u, v, g respectively. Condition 4 of Lemma 35 bounds the length of μ, ν, γ by the value $d^5 + 2d^4 + d^3 + 2d^2$ hence we

only need $O(\log(d^5)) = O(n)$ space to store these counters. The initial values of u, v and g determine which string among μ, ν and γ is longer and we start guessing the characters of such string from left to right, reducing its counter by one whenever we guess a character. When the counter reaches the same value of the second biggest counter, we start guessing the characters of both the first and the second string at the same time and we carry on until they reach the value of the last counter. Then, we guess simultaneously the characters of all three strings until all counters reach the value 0. While guessing the characters of μ (respectively, ν) we update at each step the set of states of $\mathcal{A}$ reachable from q_0 by reading the currently guessed prefix of μ (ν), so that in the end we obtain the sets $\delta(q_0, \mu)$ and $\delta(q_0, \nu)$. We proceed similarly for γ, but this time we compute the set $\delta(q, \gamma)$ for each state $q \in Q$. Since $\mathcal{D}$ is obtained by the powerset construction from $\mathcal{A}$, we can verify condition 2 of Lemma 35 by checking whether the set $\delta(q_0, \mu)$ and the set

$$\delta(q_0, \mu \cdot \gamma) = \bigcup_{p \in \delta(q_0, \mu)} \delta(p, \gamma)$$

are equal, and we do the same for $\delta(q_0, \nu)$ and $\delta(q_0, \nu \cdot \gamma)$. Condition 3 of Lemma 35 can be checked in constant space. To compare μ and γ, we use a variable ρ that indicates whether μ is less, equal or greater than γ. We initialize ρ based on the counters u, g as follows:

$$\rho := \begin{cases} = & \text{if } u = g \\ \vdash & \text{if } u < g. \end{cases}$$

We leave ρ unchanged until we start guessing simultaneously μ and γ. Then, when we guess simultaneously the character c_1 for μ and the character c_2 for γ, we set

$$\rho := \begin{cases} \prec & \text{if } c_1 \prec c_2 \\ \succ & \text{if } c_1 \succ c_2 \\ \rho & \text{if } c_1 = c_2. \end{cases}$$

Note that if at the end ρ has value $\vdash$, it means that $\mu \vdash \gamma$. Otherwise, we have $\mu \, \rho \, \gamma$. Therefore, we are always able to determine the co-lexicographic order of μ and γ and, similarly, of ν and γ. To check condition 1 of Lemma 35, consider the automata A_μ and A_ν obtained from the NFA $\mathcal{A}$ by considering as initial states the sets $\delta(q_0, \mu)$ and $\delta(q_0, \nu)$, respectively. We have that $\mu \not\equiv_{\mathcal{L}} \nu$ if and only if $\mathcal{L}(\mathcal{A}_\mu) \neq \mathcal{L}(\mathcal{A}_\nu)$, and checking whether $\mathcal{L}(\mathcal{A}_\mu) = \mathcal{L}(\mathcal{A}_\nu)$ can be done in polynomial space, since deciding whether two NFAs recognize the same language is a well-known PSPACE-complete problem.

To prove the hardness of the problem, we will show a polynomial reduction from the universality problem for NFA, i.e. the problem of deciding whether the language accepted by an NFA $\mathcal{A}$, over the alphabet Σ, is Σ^*.

Let $\mathcal{A} = (Q, q_0, \delta, F, \Sigma)$ be an NFA and let $\mathcal{L} = \mathcal{L}(\mathcal{A})$. We can assume without loss of generality that $q_0 \in F$, otherwise $\mathcal{A}$ would not accept the empty string and we could immediately derive that $\mathcal{L} \neq \Sigma^*$. Let a, b, c be three characters not in Σ and let $a \prec b \prec c$ (the order of the characters of Σ is irrelevant in this proof). First, we build the automaton $\mathcal{A}'$ starting from $\mathcal{A}$ by adding an edge (q_f, c, q_0) for each final state $q_f \in F$, see the top part of Figure 2.16. Notice that, since $\epsilon \in \mathcal{L}$, $\mathcal{A}'$ recognizes the language $\mathcal{L}' = \mathcal{L}(\mathcal{A}') = (\mathcal{L}c)^* \cdot \mathcal{L}$. We claim that $\mathcal{L} = \Sigma^*$ if and only if $\mathcal{L}' = (\Sigma \cup \{c\})^*$: if $\mathcal{L} = \Sigma^*$ and α is a string in $(\Sigma \cup \{c\})^*$, we prove that $\alpha \in \mathcal{L}'$. Let n be the number of occurrences of c in α. Then $\alpha = \alpha_0 \, c \, \alpha_2 \, c \, \ldots \, \alpha_{n-1} \, c \, \alpha_n$ for some

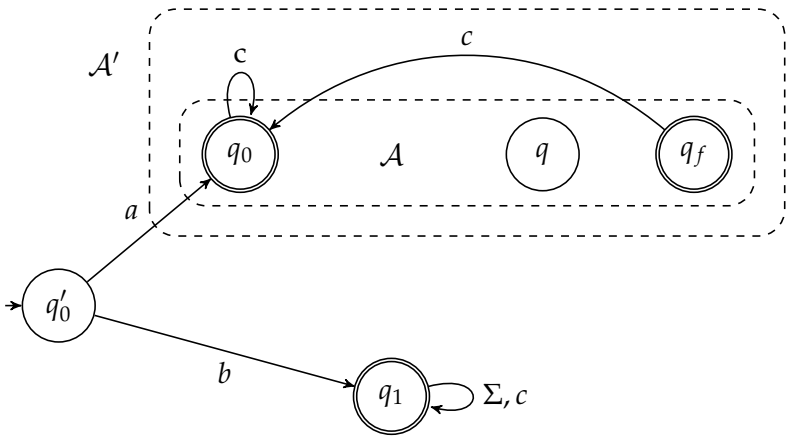


FIGURE 2.16: The automaton $\mathcal{A}''$. Every accepting state of $\mathcal{A}$, labeled q_f in the figure, has a back edge labeled c connecting it to q_0. Conversely, non-accepting states of $\mathcal{A}$, labeled q in the figure, do not have such back edges.

$\alpha_1, \ldots, \alpha_n \in \Sigma^*$. Hence $\alpha \in (\Sigma^* c)^* \cdot \Sigma^* = \mathcal{L}'$. If instead $\mathcal{L} \neq \Sigma^*$, let α be a string in $\Sigma^* \setminus \mathcal{L}$. Then $\alpha \cdot c \notin \mathcal{L}'$ by construction.

We build a second automaton $\mathcal{A}''$ by adding a non-final state q'_0, which will be the unique initial state of $\mathcal{A}''$, and a final state q_1. We also add the edges $(q'_0, a, q_0), (q'_0, b, q_1), (q_1, c, q_1)$ and, for all character $d \in \Sigma$, the self-loop (q_1, d, q_1), as depicted in Figure 2.16. Let $\mathcal{L}'' = \mathcal{L}(\mathcal{A}'')$ be the language recognized by $\mathcal{A}''$. We claim that $\mathcal{L} = \Sigma^*$ if and only if $\mathcal{L}''$ is Wheeler.

($\Longrightarrow$) If $\mathcal{L} = \Sigma^*$, we have already proved that $\mathcal{L}' = (\Sigma \cup \{c\})^*$. Hence we have $\mathcal{L}'' = \{a, b\} \cdot (\Sigma \cup \{c\})^*$. The minimum DFA recognizing $\mathcal{L}''$ has only one loop, therefore by Theorem 16 $\mathcal{L}''$ is Wheeler.

($\Longleftarrow$) If $\mathcal{L} \neq \Sigma^*$, let α be a string in $\Sigma^* \setminus \mathcal{L}$. Note that $\alpha \neq \varepsilon$ since we assumed that $\varepsilon \in \mathcal{L}$. Every possible run of α over $\mathcal{A}$ must lead to a non-accepting state, hence $\alpha \cdot c \notin \mathcal{L}'$. This implies that for all $i \geq 0$ we have $a \cdot c^i \cdot \alpha \cdot c \notin \mathcal{L}''$ (notice that the only edge labeled c leaving q_0 ends in q_0). On the other hand, for all $j \geq 0$ we have $bc^j \cdot \alpha \cdot c \in \mathcal{L}''$, hence for all $i, j \geq 0$ we have $ac^i \not\equiv_{\mathcal{L}''} bc^j$. Thus the following monotone sequence in $\mathrm{Pref}(\mathcal{L}'')$

$$ac \prec bc \prec acc \prec bcc \prec \cdots \prec ac^n \prec bc^n \prec \ldots$$

is not eventually constant modulo $\equiv_{\mathcal{L}''}$. From Lemma 15 it follows that $\mathcal{L}''$ is not Wheeler.

Notice that in the reduction described in Figure 2.16, if the starting NFA $\mathcal{A}$ was reduced, then also $\mathcal{A}''$ would be reduced. This means that the statement of Theorem 36 holds even if restricted to reduced NFAs. □

Remark 37. Note that the previous theorem is in contrast with what happens when we consider the problem of deciding whether an NFA is Wheeler, instead of whether it accepts a Wheeler language: in the former case, restricting the problem to reduced NFAs makes it solvable in polynomial time.

2.5 Conclusions

In this chapter we considered a number of computational complexity problems related with the general idea of ordering states of a finite automaton. In general, ordering objects might lead to significant simplification of otherwise difficult storage and/or manipulation problems. In fact, ordered finite automata can ease such tasks as index construction, membership testing, and even determinization of NFAs accepting a given regular language. Clearly, a key point is the complexity of finding a right order from scratch. Even though finding such an order turned out to be simple on DFAs and turning a Wheeler NFA into a Wheeler DFA is polynomial, in contrast to the *un*-ordered case, things become much more tricky when the input automaton is a non-deterministic one. This issue, together with some of its natural variants, were the main theme of this chapter. We proved that a number of ordered-related results, ultimately guaranteeing the existence of polynomial time algorithms on DFAs, are much more complex if the starting automaton is an NFA —even in the case of reduced NFAs.

A problem that this chapter leaves open concerns the optimal complexity of an algorithm that computes the minimum WDFA starting from a fingerprint of a language. The algorithm we proposed in Proposition 23 is almost optimal: except for a factor of n (equal to the upper bound UB and coinciding with the size of the input) which can be considered negligible when the size of the output m is exponential in n —that is, when the problem is truly difficult— the complexity is equal to $m \log m$. The $\log m$ factor, which is the last obstacle to achieving optimal complexity, is due to the fact that the fingerprint provided as input in the Proposition is completely generic; therefore, it is necessary to use binary search to determine the edges of the minimum WDFA. However, as we have already noted, Algorithm 1 does not provide just any fingerprint but one with particular characteristics; perhaps it is possible to exploit these characteristics to determine the edges of the minimum WDFA more quickly, thus eliminating the $\log m$ factor.

Chapter 3

Decomposing Wheeler automata

In this section we will describe the general idea of the Krohn-Rhodes Decomposition Theorem, a useful tool that allows us to split a DFA into simpler pieces, connected by a cascade product (see Definition 10). More in general, the cascade decomposition applies to semiautomata, that is, DFAs where initial and final states are not specified.

We start with an example illustrating a decomposition of the input-consistent semiautomaton $\mathcal{D} = (Q, \delta, \Sigma)$ in Figure 3.1. The input-consistency of $\mathcal{D}$ highlights an underlying structure of the semiautomaton: we can collapse all the states with the same input-character into superstates—one for each character of the alphabet—to obtain a smaller (and coarser) semiautomaton that mimics the transitions of the original semiautomaton, see Figure 3.2. The semiautomaton $\mathcal{D}_1 = (Q^1, \delta^1, \Sigma)$ is a compact version of $\mathcal{D}$: we can retrieve all the possible runs of $\mathcal{D}$ but with less details. For instance the run of the string *aba* over $\mathcal{D}$, that is $1, 2, 3, 4$, becomes $\{1,3\}, \{2,4\}, \{1,3\}, \{2,4\}$ in $\mathcal{D}_1$. We can consider the semiautomaton $\mathcal{D}_1$ as a *first approximation* of $\mathcal{D}$ from which, however, we may retrieve the original semiautomaton starting from this approximation by using the cascade product. That is, it is possible to build a second semiautomaton $\mathcal{D}_2$ such that $\mathcal{D} = \mathcal{D}_1 \circ \mathcal{D}_2$. How many states should $\mathcal{D}_2$ have? Since the set of states of the cascade product $\mathcal{D}_1 \circ \mathcal{D}_2$ is the direct product of the set of states of $\mathcal{D}_1$ and $\mathcal{D}_2$ and the superstates of $\mathcal{D}_1$ were both obtained by collapsing two states of $\mathcal{D}$, the semiautomaton $\mathcal{D}_2$ only needs two states to differentiate the possible behaviours of the states of $\mathcal{D}$ included into the superstates of $\mathcal{D}_1$: the transitions of $\mathcal{D}_2$, labeled by characters in $Q^1 \times \Sigma$, will take care of the rest. More precisely, we let $\{A, B\}$ be the set of states of $\mathcal{D}_2$ and we proceed as follows: for the superstate of $\mathcal{D}_1$ labeled $\{1,3\}$ we arbitrarily associate the state 1 to A and the state 3 to B. Analogously, for the second superstate labeled $\{2,4\}$ we arbitrarily associate the state 4 to A and the state 2 to B. Now, in order to retrieve the original semiautomaton as the cascade product of $\mathcal{D}_1$ and $\mathcal{D}_2$, we need to add the correct edges to $\mathcal{D}_2$. The possible labels of the edges are $(\{1,3\}, a), (\{1,3\}, b), (\{2,4\}, a), (\{2,4\}, b)$ and we assign them according to the labels A and B that we associated to states of $\mathcal{D}_1$. For instance, consider the state A of $\mathcal{D}_2$: if we read the transition $(\{1,3\}, a)$, which state should we reach? In the superstate $\{1,3\}$, the state A is associated to state 1, hence from 1 we follow transition a in the original semiautomaton ending

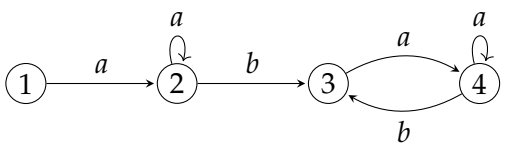


FIGURE 3.1: An input-consistent semiautomaton $\mathcal{D}$.

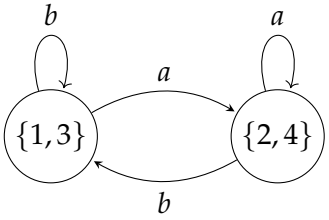


FIGURE 3.2: The semiautomaton $\mathcal{D}_1$, a first approximation of the semiautomaton $\mathcal{D}$ depicted in Figure 3.1.

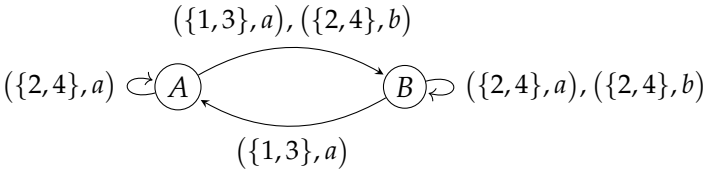


FIGURE 3.3: The automaton $\mathcal{D}_2$, complementing the automaton $\mathcal{D}_1$.

up in state 2. In parallel, we also follow the transition a in $\mathcal{D}_1$ applied to the superstate $\{1,3\}$, ending up in the supestate $\{2,4\}$. Since, for the superstate $\{2,4\}$, we associated the state 2 to B, we finally elect B as the state reached from A reading $(\{1,3\},a)$. All other possible transitions are determined in a similar manner, see Figure 3.3. The reader can check that, letting $x \in \{\{1,3\},\{2,4\}\}$ and $y \in \{A,B\}$, by identifying a state (x,y) of the cascade product of $\mathcal{D}_1$ and $\mathcal{D}_2$ with the $\mathcal{D}$ state in x associate to y, we obtain a DFA which is isomprphic to $\mathcal{D}$.

Summarizing, we started from a semiautomaton with 4 states and we managed to "disassemble" it into two smaller semiautomata: a first approximation $\mathcal{D}_1$ and a second semiautomaton correcting the not-completly-faithful behaviour of the first one. This is a nice starting point for a decomposition of semiautomata: we start from a complex semiautomaton $\mathcal{D}$ and we end up with two smaller (simpler?) semiautomata from which is possible to retrieve $\mathcal{D}$ using the cascade product.

Now a natural question is: are $\mathcal{D}_1$ and $\mathcal{D}_2$ really simpler than $\mathcal{D}$? There should be no doubt about $\mathcal{D}_1$: it is an approximation of $\mathcal{D}$ which, in fact, loses some information. What about $\mathcal{D}_2$? Is the condition of having less states enough to guarantee simplification? The answer to this question is, unfortunately, no. This is due to the fact that, in addition to the number of $\mathcal{D}_2$'s states, we must look at the "nature" of its transitions. That is, the entire collection of transitions labeled by c must be analysed, for each $c \in \Sigma$. Formally, given a semiautomaton $\mathcal{D} = (Q, \delta, \Sigma)$, the *action* of $c \in \Sigma$ over Q is the function $\delta^c : Q \to Q$

$$\delta^c(q) = q' \iff \delta(q,c) = q'.$$

The *spectrum* of possible non-trivial (i.e. different from the identity) actions over a set, has two "extremes": actions that send all the elements to the same one—to be called *resets*—and actions that send distinct elements in distinct elements—to be called *permutations*. The opposite nature of this two kind of actions resides in the amount of information that they carry over. Given a permutation δ for which $\delta(x) = q'$ is known, we can uniquely determine x. If instead we are given a reset δ and we know that $\delta(x) = q'$, we have no information about what x could be: all the elements of Q are eligible for such a choice. These two extremes will be used to measure the complexity of our transitions that will go from the simplest one (resets) to

the most complex ones (permutations). We will also classify "basic" semiautomata accordingly: the very simple reset semiautomata will be the ones where every transition is either a reset or the identity, while the complex permutation semiautomata will be the ones where every transition is a permutation—possibly the identity.

If we look back at the decomposition proposed at the beginning of this section, we notice that the first semiautomaton is a reset whereas the second still contains a permutation —labeled $(\{1,3\},a)$. However, the original DFA $\mathcal{D}$ did not contain any permutation: hence we can conclude that the resulting composition introduced a grade of complexity not needed! This is, in fact, the case and it is indeed possible, by swapping the initial labelling choice of A and B in the superstate $\{2,4\}$, to obtain a reset semiautomaton as $\mathcal{D}_2$, see Figure 3.4.

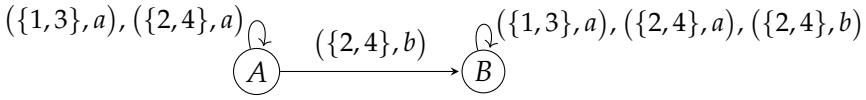


FIGURE 3.4: The semiautomaton $\mathcal{D}_2'$, obtained when swapping the assignment of A, B to the states in $\{2,4\}$. Note that here all transitions are either resets or the identity.

In the Krohn-Rhodes Decomposition Theorem (KRDT) it is shown how to decompose any semiautomaton in such a way that the number of permutation components is as small as possible. In particular, if the semiautomaton is counter-free[1] then the decomposition is permutation-free. To obtain such a result, a careful choice of the initial partition[2] and a careful assignment of the labels of the second automaton are needed, resulting in a computationally heavy algorithm [38] (see [39] for an implementation). However, it is worth the effort: the applications related to the Krohn-Rhodes decomposition are numerous, ranging from biology to physics to psychology and beyond [40].

In this thesis we will focus on the decomposition of Wheeler automata, answering the following question: is there a way to exploit the Wheeler state order to easily obtain a feasible —computationally light— decomposition that does not produce unwelcome permutation components?[3]

3.1 Cascade product and the Krohn-Rhodes Decomposition Theorem

The KRDT is a theorem on deterministic finite automata (DFAs), but initial and final states play no role in its statement. Hence we start defining semiautomata, which are DFAs where initial and final states are not specified. Moreover, the general KRDT theorem is stated for complete deterministic semiautomata, where transitions from any states are always defined for every character. The path coherent and Wheeler

[1] Being counter-free does not depend on initial or final states of the automaton, hence it makes sense to consider counter-free semiautomaton.

[2] Actually, in our example we started from an input-consistent DFA and a partition of its set of states in order to have a simple example, but in general KRT over (possibly non input-consistent) DFA will use a cover of the set of states, where elements may overlap, see Section **??**.

[3] Notice that a permutation-free decomposition of a Wheeler automaton is already granted by KRDT, but computing it is an hard task.

automata are inherently partial, since their classes are not closed under complementation. Hence, we introduce the class of partial semiautomata and we state KRDT for this class.

A partial semiautomaton is a tuple $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, \delta^{\mathcal{A}})$ where: (i) $Q^{\mathcal{A}}$ is a finite set of states; (ii) $\Sigma^{\mathcal{A}}$ is the alphabet; (iii) $\delta^{\mathcal{A}} : Q^{\mathcal{A}} \times \Sigma^{\mathcal{A}} \to Q^{\mathcal{A}}$ is a partial function. From here onwards, by a semiautomaton we always mean a partial semiautomaton. For any $a \in \Sigma^{\mathcal{A}}$, we denote by $\delta^{\mathcal{A}}(-, a)$ the (possibly partial) function from $Q^{\mathcal{A}}$ to $Q^{\mathcal{A}}$ that maps q in $\delta^{\mathcal{A}}(q, a)$, for each $q \in Q^{\mathcal{A}}$. Moreover, for any subset of states $Q \subseteq Q^{\mathcal{A}}$ and any symbol $a \in \Sigma^{\mathcal{A}}$, we denote by $\delta^{\mathcal{A}}(Q, a)$ the set $\{\delta^{\mathcal{A}}(q, a) \mid q \in Q\}$. Since we will also introduce semiautomata $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, \delta^{\mathcal{A}})$ and $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma^{\mathcal{B}}, \delta^{\mathcal{B}})$ where $\Sigma^{\mathcal{A}} = \Sigma^{\mathcal{B}}$ and states in $Q^{\mathcal{B}}$ are subsets of $Q^{\mathcal{A}}$ – that is, $Q^{\mathcal{B}} \subseteq Pow(Q^{\mathcal{A}})$ –, we avoid ambiguity by always explicitly indicating the automaton, as in $\delta^{\mathcal{A}}(Q, a)$ and $\delta^{\mathcal{B}}(Q, a)$, so that it is clear whether Q is a state of $\mathcal{B}$ or a subset of $\mathcal{A}$-states.

Finally, if (q, a) does *not* belong to the domain of $\delta^{\mathcal{A}}$ we write $\delta^{\mathcal{A}}(q, a) = \perp$, where $\perp \notin Q^{\mathcal{A}}$. More in general, for a partial function $f : X \to Y$ and an element $x \in X$ we write $f(x) = \perp$ to denote that $x \notin \mathrm{dom}(f)$. Clearly, a function can be defined only if all of its arguments are defined, thus we consider every function that has $\perp$ as one of its arguments as undefined – e.g. $f(\perp) = \perp$ and $(\perp, x) = (x, \perp) = \perp$ for all x.

Notice that to each semiautomaton $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, \delta^{\mathcal{A}})$ we can associate a semigroup generated by the actions $\delta^{\mathcal{A}}(-, a)$ over the set $Q^{\mathcal{A}}$. This semigroup is called the *transition semigroup* of $\mathcal{A}$ and it plays an important role in the algebraic theory of automata (see, among others, [41]).

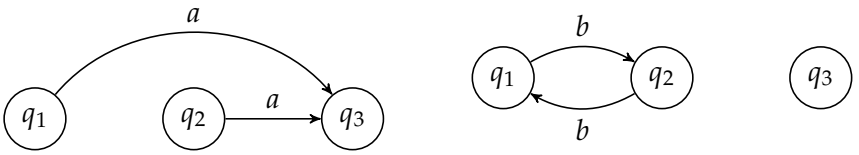
Definition 19. A transition $a \in \Sigma$ in a semiautomaton $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, \delta^{\mathcal{A}})$ is:

1) an *identity* if $\delta^{\mathcal{A}}(q, a) \in \{\perp, q\}$ for all $q \in Q^{\mathcal{A}}$;

2) a *reset* if there exists a state $r \in Q^{\mathcal{A}}$ such that $\delta^{\mathcal{A}}(q, a) \neq \perp$ implies $\delta^{\mathcal{A}}(q, a) = r$, for all $q \in Q^{\mathcal{A}}$;

3) a *permutation* if the function $\delta^{\mathcal{A}}(-, a)$ is a permutation over its domain i.e. it is injective and maps its domain onto itself.

A semiautomaton $\mathcal{A}$ is a permutation-reset semiautomaton if all its transitions are permutations or resets. We say that $\mathcal{A}$ is a reset semiautomaton if all its transitions are either resets or identities. We say that $\mathcal{A}$ is a permutation automaton if all its transitions are permutations.



(a) The transition a is a *reset*. (b) The transition b is a *permutation*.

FIGURE 3.5: An example of *reset* and *permutation* transitions.

In the example of a decomposition given at the beginning of this section the initial automaton ends up being isomorphic to the cascade product of its components. In general, this is not always the case: the cascade product will be usually "larger" but we can retrieve the initial DFA as an *homomorphic image* of the cascade, defined below.

Definition 20 (Homomorphic image)**.** Let $\mathcal{A}, \mathcal{B}$ be two semiautomata over the same alphabet Σ. We say that $\mathcal{A}$ is an *homomorphic image* of $\mathcal{B}$ if there is a surjective (total) function $\phi\colon Q^{\mathcal{B}} \to Q^{\mathcal{A}}$ such that, for all $q \in Q^{\mathcal{B}}$ and $a \in \Sigma$,

$$\phi(\delta^{\mathcal{B}}(q,a)) = \delta^{\mathcal{A}}(\phi(q),a).$$

If ϕ is a bijection, $\mathcal{A}$ and $\mathcal{B}$ are said to be *isomorphic*.

In particular notice that, since ϕ is total on $Q^{\mathcal{A}}$ and we agreed that $\phi(\perp) = \perp$, if $q' \in Q^{\mathcal{A}}$ and $q \in \phi^{-1}(q')$ then $\delta^{\mathcal{B}}(q,a) = \perp$ iff $\delta^{\mathcal{A}}(q',a) = \perp$.

A semiautomaton $\mathcal{A}$ can be transformed into a language acceptor by fixing an initial state s and a set of final states F and setting, as usual, $\mathcal{L}(\mathcal{A}) = \{\alpha \in (\Sigma^{\mathcal{A}})^* \mid \delta^{\mathcal{A}}(s,\alpha) \in F\}$. We say that a semiautomaton $\mathcal{B}$ is at least as expressive as the semiautomaton $\mathcal{A}$ if any language accepted by $\mathcal{A}$ is also accepted by $\mathcal{B}$. The next lemma proves that if $\mathcal{A}$ is a homomorphic image of $\mathcal{B}$ then $\mathcal{B}$ is at least as expressive as $\mathcal{A}$.

Lemma 38. *If the semiautomaton $\mathcal{A}$ is a homomorphic image of the semiautomaton $\mathcal{B}$ and the language $\mathcal{L}$ is accepted by $\mathcal{A}$, then $\mathcal{L}$ is also accepted by $\mathcal{B}$.*

Proof. Let $\phi\colon Q^{\mathcal{B}} \to Q^{\mathcal{A}}$ be an homomorphism between the semiautomaton $\mathcal{A}$ and $\mathcal{B}$. Let $s \in Q^{\mathcal{A}}$ and $F \subseteq Q^{\mathcal{A}}$ be the initial state and the final states, respectively, of the automaton $(Q^{\mathcal{A}}, s, \delta^{\mathcal{A}}, F)$ and let $\mathcal{L}$ be the language recognized by $\mathcal{A}$:

$$\mathcal{L} = \{\alpha \in \Sigma^* \mid \delta^{\mathcal{A}}(s,\alpha) \in F\}.$$

Let $s' \in Q^{\mathcal{B}}$ be such that $\phi(s') = s$ and let $F' = \phi^{-1}(F)$. Then one can prove by induction on the length of the string α that

$$\delta^{\mathcal{B}}(s',\alpha) \in F' \Leftrightarrow \delta^{\mathcal{A}}(s,\alpha) \in F,$$

which proves that the language recognized by the two DFAs is the same. □

Next, we state some basic results about cascades and homomorphic images necessary to state KRDT.

Proposition 39.

1) *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be semiautomata such that the cascade product $(\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C}$ is defined. In particular, $\Sigma^{\mathcal{C}} = Q^{\mathcal{A}} \times Q^{\mathcal{B}} \times \Sigma^{A}$. Then, the semiautomaton $\mathcal{C}' = (Q^{\mathcal{C}'}, \Sigma^{\mathcal{C}'}, \delta^{\mathcal{C}'})$ obtained from $\mathcal{C}$ by renaming, for each edge, the label $(q^{\mathcal{A}}, q^{\mathcal{B}}, a)$ as $(q^{\mathcal{B}}, q^{\mathcal{A}}, a)$, formally defined as:*

$$\begin{aligned} &Q^{\mathcal{C}'} := Q^{\mathcal{C}};\\ &\Sigma^{\mathcal{C}'} := Q^{\mathcal{B}} \times Q^{\mathcal{A}} \times \Sigma^{A};\\ &\delta^{\mathcal{C}'}\left(q^{\mathcal{C}}, (q^{\mathcal{B}}, q^{\mathcal{A}}, a)\right) := \delta^{\mathcal{C}}\left(q^{\mathcal{C}}, (q^{\mathcal{A}}, q^{\mathcal{B}}, a)\right), \end{aligned}$$

   is such that $(\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C} = \mathcal{A} \circ (\mathcal{B} \circ \mathcal{C}')$.

   Equivalently, if $\mathcal{A} \circ (\mathcal{B} \circ \mathcal{C})$ is defined, then there exist a semiautomaton $\mathcal{C}'$ such that $\mathcal{A} \circ (\mathcal{B} \circ \mathcal{C}) = (\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C}'$. The semiautomaton $\mathcal{C}'$ is again obtained from $\mathcal{C}$ by renaming the labels of the edges of $\mathcal{C}$.

2) *If $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ are semiautomata such that $\mathcal{A}$ is an homomorphic image of $\mathcal{B} \circ \mathcal{C}$ and $\mathcal{C}$ is an homomorphic image of $\mathcal{D}$, then $\mathcal{A}$ is an homomorphic image of $\mathcal{B} \circ \mathcal{D}$.*

Proof.
1) Let $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, \delta^{A}), \mathcal{B} = (Q^{\mathcal{B}}, Q^{\mathcal{A}} \times \Sigma^{\mathcal{A}}, \delta^{\mathcal{B}}), \mathcal{C} = (Q^{\mathcal{C}}, Q^{\mathcal{A}} \times Q^{\mathcal{B}} \times \Sigma^{\mathcal{A}}, \delta^{\mathcal{C}})$ be three semiautomata, and let $\mathcal{C}'$ be the semiautomaton defined as in the statement of the proposition. In the following, the letter u, v, t will denote a generic state of $\mathcal{A}, \mathcal{B}, \mathcal{C}$ respectively, and a will denote a generic character in $\Sigma^{\mathcal{A}}$.
Let $\mathcal{A} \circ \mathcal{B}$ be the cascade product of $\mathcal{A}$ and $\mathcal{B}$. By definition, we have

$$\delta^{\mathcal{A}\circ\mathcal{B}}\Big((u,v),a\Big) = \Big(\delta^{\mathcal{A}}(u,a), \delta^{\mathcal{B}}\big(v,(u,a)\big)\Big). \tag{3.1}$$

Let $(\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C}$ be the cascade product of $\mathcal{A} \circ \mathcal{B}$ and $\mathcal{C}$. By definition, we have

$$\delta^{(\mathcal{A}\circ\mathcal{B})\circ\mathcal{C}}\Big((u,v,t),a\Big) = \Big(\delta^{\mathcal{A}\circ\mathcal{B}}\Big((u,v),a\Big), \delta^{\mathcal{C}}\Big(t,(u,v,a)\Big)\Big). \tag{3.2}$$

Let $\mathcal{B} \circ \mathcal{C}'$ be the cascade product of $\mathcal{B}$ and $\mathcal{C}'$. By definition, we have

$$\delta^{\mathcal{B}\circ\mathcal{C}'}\Big((v,t),(u,a)\Big) = \Big(\delta^{\mathcal{B}}\Big(v,(u,a)\Big), \delta^{\mathcal{C}'}\Big(t,(v,u,a)\Big)\Big). \tag{3.3}$$

Let $\mathcal{A} \circ (\mathcal{B} \circ \mathcal{C}')$ be the cascade product of $\mathcal{A}$ and $\mathcal{B} \circ \mathcal{C}'$. By definition, we have

$$\delta^{\mathcal{A}\circ(\mathcal{B}\circ\mathcal{C}')}\big((u,v,t),a\big) = \Big(\delta^{\mathcal{A}}(u,a), \delta^{\mathcal{B}\circ\mathcal{C}'}\Big((v,t),(u,a)\Big)\Big). \tag{3.4}$$

By substituting (3.1) in (3.2), we obtain that

$$\delta^{(\mathcal{A}\circ\mathcal{B})\circ\mathcal{C}}\Big((u,v,t),a\Big) = \Big(\delta^{\mathcal{A}}(u,a), \delta^{\mathcal{B}}\Big(v,(u,a)\Big), \delta^{\mathcal{C}}\Big(t,(u,v,a)\Big)\Big).$$

Similarly, by substituting (3.3) in (3.4) we obtain that

$$\delta^{\mathcal{A}\circ(\mathcal{B}\circ\mathcal{C}')}\Big((u,v,t),a\Big) = \Big(\delta^{\mathcal{A}}(u,a), \delta^{\mathcal{B}}\Big(v,(u,a)\Big), \delta^{\mathcal{C}'}\Big(t,(v,u,a)\Big)\Big).$$

Since, by hypothesis,

$$\delta^{\mathcal{C}'}\big(t,(v,u,a)\big) = \delta^{\mathcal{C}}\big(t,(u,v,a)\big),$$

we have $(\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C} = \mathcal{A} \circ (\mathcal{B} \circ \mathcal{C}')$.

2) In the following, the letter u, v, t, x will denote a generic state of $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ respectively, and a will denote a generic character in $\Sigma = \Sigma^{\mathcal{A}} = \Sigma^{\mathcal{B}}$. Notice that, since the products $\mathcal{B} \circ \mathcal{C}$ and $\mathcal{B} \circ \mathcal{D}$ are defined, it holds $\Sigma^{\mathcal{C}} = \Sigma^{\mathcal{D}} = Q^{\mathcal{B}} \times \Sigma^{\mathcal{B}}$. By definition of $\mathcal{B} \circ \mathcal{C}$ we have

$$\delta^{\mathcal{B}\circ\mathcal{C}}\Big((v,t),a\Big) = \Big(\delta^{\mathcal{B}}(v,a), \delta^{\mathcal{C}}\big(t,(v,a)\big)\Big). \tag{3.5}$$

By definition of $\mathcal{B} \circ \mathcal{D}$ we have

$$\delta^{\mathcal{B}\circ\mathcal{D}}\Big((v,x),a\Big) = \Big(\delta^{\mathcal{B}}(v,a), \delta^{\mathcal{D}}\big(x,(v,a)\big)\Big). \tag{3.6}$$

Since $\mathcal{A}$ is an homomorphic image of $\mathcal{B} \circ \mathcal{C}$, there is an homomorphism $\phi_1 : Q^{\mathcal{B}} \times Q^{\mathcal{C}} \to Q^{\mathcal{A}}$ such that

$$\phi_1\Big(\delta^{\mathcal{B}\circ\mathcal{C}}\big((v,t),a\big)\Big) = \delta^{\mathcal{A}}\Big(\phi_1(v,t),a\Big). \tag{3.7}$$

Similarly, since $\mathcal{C}$ is an homomorphic image of $\mathcal{D}$, there is an homomorphism $\phi_2 : Q^{\mathcal{D}} \to Q^{\mathcal{C}}$ such that

$$\phi_2\Big(\delta^{\mathcal{D}}\big(x,(v,a)\big)\Big) = \delta^{\mathcal{C}}\Big(\phi_2(x),(v,a)\Big). \tag{3.8}$$

Define the homomorphism $\phi\colon Q^{\mathcal{B}\circ\mathcal{D}} \to Q^{\mathcal{A}}$ as

$$\phi(v,x) := \phi_1\Big(v,\phi_2(x)\Big), \tag{3.9}$$

we claim that $\mathcal{A}$ is an homomorphic image of $\mathcal{B}\circ\mathcal{D}$ through ϕ. In fact, we have

$$\begin{aligned}
&\phi\left(\delta^{\mathcal{B}\circ\mathcal{D}}\Big(\big(v,x\big),a\Big)\right) \overset{(3.6)}{=} \phi\left(\Big(\delta^{\mathcal{B}}(v,a),\delta^{\mathcal{D}}\big(x,(v,a)\big)\Big)\right)\\
&\overset{(3.9)}{=} \phi_1\left(\delta^{\mathcal{B}}(v,a),\phi_2\Big(\delta^{\mathcal{D}}\big(x,(v,a)\big)\Big)\right)\\
&\overset{(3.8)}{=} \phi_1\left(\delta^{\mathcal{B}}(v,a),\delta^{\mathcal{C}}\Big(\phi_2(x),(v,a)\Big)\right)\\
&\overset{(3.5)}{=} \phi_1\left(\delta^{\mathcal{B}\circ\mathcal{C}}\Big(\big(v,\phi_2(x)\big),a\Big)\right) \overset{(3.7)}{=} \delta^{\mathcal{A}}\Big(\phi_1\big(v,\phi_2(x)\big),a\Big)\\
&\overset{(3.9)}{=} \delta^{\mathcal{A}}\Big(\phi(v,x),a\Big).
\end{aligned}$$

□

Remark 40. Note that statement 1) of Proposition 39 is very close to saying that the cascade product is associative. The automata $\mathcal{C}$ and $\mathcal{C}'$, although not isomorphic as they are defined over different alphabets, are in fact the same automaton: it is sufficient to rename the labels of the edges to transform the first automaton into the second and vice versa. To achieve true associativity, [25] uses a more general definition of the cascade product, which essentially allows for the renaming of labels at will when performing the product. To avoid overloading the notation, throughout this chapter we will continue to use Definition 10 of the cascade product. Therefore, whenever we write an expression like $\mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$, we will assume that the parentheses are left-associated. Nonetheless, it is important to keep in mind that, in all the subsequent results, it is always possible to assume the parentheses are associated at will, provided the more general definition of the cascade product as defined in [25] is used.

The general KRDT states that each semiautomaton $\mathcal{A}$ is a homomorphic image of a sub-semiautomaton of a cascade product of two-state resets and permutation automata. Moreover, the cascade can be chosen in such a way that, for each permutation semiautomaton in the cascade, the semigroup generated by the transitions of the semiautomaton is a simple group which is an homomorphic image of subgroups of the semigroup generated by the $\mathcal{A}$-transitions.

Theorem 41 (KRDT)**.** *Let $\mathcal{A}$ be a deterministic partial semiautomon. Then, there are $\mathcal{B}_1,\ldots,\mathcal{B}_k$ which are either permutation or (two-states) reset semiautomata such that:*

1. *$\mathcal{A}$ is an homomorphic image of a sub-semiautomaton[4] of the cascade product $\mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$;*
2. *if $\mathcal{B}_i$ is a permutation automaton, then its transition semigroup is a subgroup of the transition semigroup of $\mathcal{A}$.*

[4] That is, a subset of states closed under transitions.

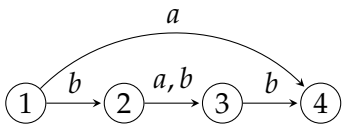


FIGURE 3.6: Example of a path coherent semiautomaton.

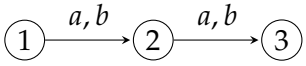


FIGURE 3.7: A path coherent semiautomaton not satisfying (W1).

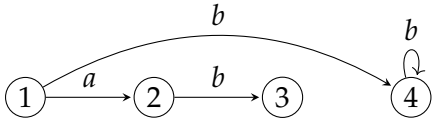


FIGURE 3.8: A path coherent semiautomaton satisfying (W1) but not (W2).

Since the transition semigroup of a counter-free automaton is group-free (see [19]), the second condition of KRDT implies that counter-free DFAs can be decomposed into a cascade of reset automata only[5]. Since Wheeler automata are counter-free, the same result applies to them. However, in the following section we will improve this result, exploiting the order of states of Wheeler automata, in two ways. First, we will prove that the number of components needed for the decomposition is at most n, whereas inspecting the KRDT proof we obtain a bound of $n \log n$ even in the counter-free case. Second, the proof will be much simpler than the one given in the literature for the counter-free case.

3.2 Krohn-Rhodes Decomposition Theorem for Wheeler automata

In this section we give a simpler proof of KRDT for the path coherent (see Definition 21) and Wheeler class. In the Wheeler case we also prove that the numbers of blocks –two-state resets– is linear in the number of states of the original automaton.

Definition 21 (Path Coherence)**.** Let $<$ be a total order over the set of states S^A of a semiautomaton $A = (S^A, \Sigma^A, \delta^A)$. Then $(A, <)$ is *path coherent* if, for all $a \in \Sigma^A$, the function $\delta(-, a)$ maps intervals in intervals. Equivalently, $(A, <)$ is *path coherent* if, for all $q < q' \in S^A$, $a \in \Sigma^A$, and $x, y, z \in S^A$ it holds that:

$$x < z < y \ \wedge \ x, y \in \delta^A([q, q'], a) \ \rightarrow \ z \in \delta^A([q, q'], a).$$

Figure 3.6 shows an example of a path coherent automaton on the alphabet $\Sigma = \{a, b\}$. Notice that (W2) is not satisfied.

In general, a path coherent semiautomaton may not satisfy condition (W1) of Definition 6, for any order of the alphabet. Consider for example the semiautomaton in Figure 3.7. Moreover, path coherence and (W1) do not necessarily imply (W2). Consider for example the semiautomaton in Figure 3.8.

However, we do have the following dependencies.

[5]The general KRDT does not prove that the original automaton is a homomorphic image of the cascade: one has to go through a sub-semiautomaton of the cascade. For path consistent and Wheeler class we can avoid the use of sub-semiautomata.

Proposition 42. *1) If a semiautomaton $\mathcal{A}$ satisfies both path coherence and input-consistency then there is an order $\prec$ over the alphabet Σ for which $\mathcal{A}$ satisfies (W1). In particular, a path coherent, input-consistent semiautomaton satisfying (W2) is a Wheeler semiautomaton. 2) Any semiautomaton $\mathcal{A}$ satisfying both (W1) and (W2) and such that any state has at least an incoming transition is path coherent.*

Proof. 1. For any letter $a \in \Sigma$, path coherence implies that the set of states with an ingoing a-transition, i.e. $\delta(Q^{\mathcal{A}}, a)$, is an interval, whereas input consistency implies that if $a \neq b$ then $\delta(Q^{\mathcal{A}}, a) \cap \delta(Q^{\mathcal{A}}, b) = \emptyset$. Hence we may define $a \prec b$ iff the interval $\delta(Q^{\mathcal{A}}, a)$ precedes the interval $\delta(Q^{\mathcal{A}}, b)$ in the order $<$, satisfying in this way (W1).

2. Suppose $q < q' \in Q^{\mathcal{A}}$, $a \in \Sigma$, $x, y, z \in Q^{\mathcal{A}}$, and $x < z < y$ for $x, y \in \delta^{\mathcal{A}}([q, q'], a)$. Let $r, t \in [q, q']$ be such that $x = \delta^{\mathcal{A}}(r, a), y = \delta^{\mathcal{A}}(t, a)$ and let $b \in \Sigma, w \in Q^{\mathcal{A}}$ be such that $z = \delta^{\mathcal{A}}(w, b)$. Then (W1) implies $b = a$ and (W2) implies $w \in [q, q']$.

□

We are ready to show how to obtain, in a simple way, the Krohn-Rhodes decomposition of a path coherent automaton. We start by defining the first element of the cascade using the notion of *admissible decomposition* (see [25, 26]).

Definition 22. A family $\mathcal{D}$ of subsets of the set of states of a semiautomaton $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, \delta^{\mathcal{A}})$ is said to be an *admissible decomposition* if:

1. $\cup \mathcal{D} = Q^{\mathcal{A}}$;
2. for any $a \in \Sigma^{\mathcal{A}}$ the image of an element of $\mathcal{D}$ under $\delta^{\mathcal{A}}(-, a)$ is contained in at least one element of $\mathcal{D}$:

$$\forall a \in \Sigma^{\mathcal{A}}\ \forall D \in \mathcal{D}\ \exists D' \in \mathcal{D}\ \delta^{\mathcal{A}}(D, a) \subseteq D'.$$

Given an admissible decomposition $\mathcal{D}$ of a semiautomata $\mathcal{A}$ we can build a *factor* semiautomaton $\mathcal{A}/\mathcal{D}$ over the same alphabet with $Q^{\mathcal{A}/\mathcal{D}} = \mathcal{D}$ and $\delta^{\mathcal{A}/\mathcal{D}}(D, a) = D'$ where $D' \in \mathcal{D}$ is such that $\delta^{\mathcal{A}}(D, a) \subseteq D'$ (if there is more than one, then choose one arbitrarily).[6]

Given a semiautomaton $\mathcal{A}$, the first element of a cascade decomposition for $\mathcal{A}$ can be choosen to be a factor of $\mathcal{A}$ (see [25, 26]). However, for the full KRDT, which proves that there are cascades where the permutation-blocks are homomorphic images of subgroups of the transition monoid of the original automaton, one has to choose a particularly well behaved decomposition of $\mathcal{A}$.

In the following lemma and theorem we prove that, in case of path coherent semiautomata, a particular choice for the decomposition of $\mathcal{A}$ allows to use a simple induction for obtaining the full decomposition. Moreover, when applied to the Wheeler case, we shall see that this decomposition avoid altogether the use of permutations in the cascade.

Lemma 43. *Let $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, \delta^{\mathcal{A}}, <)$ be a path coherent semiautomaton with $n = |Q^{\mathcal{A}}| > 2$. Then, there are a permutation-reset semiautomaton $\mathcal{B}$ with $|Q^{\mathcal{B}}| = 2$ and a path coherent semiautomaton $\mathcal{C}$ with $|Q^{\mathcal{C}}| = n - 1$, such that $\mathcal{A}$ is a homomorphic image of $\mathcal{B} \circ \mathcal{C}$.*

[6] For a complete semiautomaton $\mathcal{A}$, where the transition functions are total, a factor of $\mathcal{A}$ is always a homomorphic image of $\mathcal{A}$. This is not always the case for partial automata.

Proof. Consider the decomposition $\mathcal{D} = \{D_0, D_1\}$ of $Q^{\mathcal{A}}$ where

$$D_0 = \{1, \ldots, n-1\}, D_1 = \{2, \ldots, n\}.$$

We have either $\delta^{\mathcal{A}}(D_i, a) \subseteq D_0$ or $\delta^{\mathcal{A}}(D_i, a) \subseteq D_1$ (possibly both) for $i = 0, 1$: it cannot be the case that both 1 and n are in $\delta^{\mathcal{A}}(D_i, a)$ or, by path coherence, we would have $\delta^{\mathcal{A}}(D_i, a) = \{1, \ldots, n\}$ which is not possible, being $\mathcal{A}$ deterministic. In other words, the decomposition $\mathcal{D}$ is admissible. Let $\mathcal{B} = \mathcal{A}/\mathcal{D} = (Q^{\mathcal{B}}, \Sigma, \delta^{\mathcal{B}})$ be the $\mathcal{D}$-*factor* of $\mathcal{A}$, defined as follows. The set of states is $Q^{\mathcal{B}} = \{D_0, D_1\}$ and we define $\delta^{\mathcal{B}}(D_j, a) = \perp$ if and only if $\delta^{\mathcal{A}}(D_j, a) = \emptyset$. If instead $\delta^{\mathcal{A}}(D_j, a) \neq \emptyset$, then we define $\delta^{\mathcal{B}}(D_j, a) = D_0$ if $\delta^{\mathcal{A}}(D_j, a) \subseteq D_0$ (or, equivalently, if $n \notin \delta^{\mathcal{A}}(D_j, a)$), whereas we define $\delta^{\mathcal{B}}(D_j, a) = D_1$ if $\delta^{\mathcal{A}}(D_j, a) \not\subseteq D_0$ (or, equivalently, $n \in \delta^{\mathcal{A}}(D_j, a)$) and hence $\delta^{\mathcal{A}}(D_j, a) \subseteq D_1$. More succinctly, when $\delta^{\mathcal{B}}(D_j, a) \neq \perp$ we have $\delta^{\mathcal{B}}(D_j, a) = D_{\gamma(j,a)}$, where $\gamma(j, a)$ is defined as

$$\gamma(j, a) = \begin{cases} 0 & \text{if } n \notin \delta^{\mathcal{A}}(D_j, a) \\ 1 & \text{if } n \in \delta^{\mathcal{A}}(D_j, a) \end{cases}$$

for all $j \in \{0, 1\}$ and $a \in \Sigma$. Note that $\mathcal{B}$ has only two states. Hence, a transition in $\mathcal{B}$ can only be a (eventually partial) reset, the identity, or the permutation $\delta^{\mathcal{B}}(D_0, a) = D_1$, $\delta^{\mathcal{B}}(D_1, a) = D_0$. Therefore, $\mathcal{B}$ is a permutation-reset automaton.

Let $\mathcal{C} = (Q^{\mathcal{C}}, Q^{\mathcal{B}} \times \Sigma, \delta^{\mathcal{C}})$ be the semiautomaton with set of states

$$Q^{\mathcal{C}} = \{C_1, \ldots, C_{n-1}\} \quad \text{with} \quad C_i = \{(i, D_0), (i+1, D_1)\} \quad \text{for } 1 \leq i \leq n-1$$

and transition function defined as follows, for $i = 1, \ldots, n-1$ and $j = 0, 1$:

$$\delta^{\mathcal{C}}\Big(C_i, (D_j, a)\Big) = C_{\delta^{\mathcal{A}}(i+j,a) - \gamma(j,a)}. \tag{3.10}$$

Note that $\delta^{\mathcal{C}}\Big(C_i, (D_j, a)\Big)$ is well defined because it always hold

$$0 < \delta^{\mathcal{A}}(i+j, a) - \gamma(j, a) < n;$$

assume, by way of contradiction, that $\delta^{\mathcal{A}}(i+j, a) - \gamma(j, a) = n$, we must have both $\delta^{\mathcal{A}}(i+j, a) = n$ and $\gamma(j, a) = 0$. Note that $i + j \in D_j$, for all $i = 1, \ldots, n-1$ and $j = 0, 1$, thus, from $\delta^{\mathcal{A}}(i+j, a) = n$ it follows that $n \in \delta^{\mathcal{A}}(D_j, a)$. By definition of γ, this means that $\gamma(j, a) = 1$: a contradiction. Therefore $\delta^{\mathcal{A}}(i+j, a) - \gamma(j, a) \neq n$. The fact that $\delta^{\mathcal{A}}(i+j, a) - \gamma(j, a) \neq 0$ can be proved in a similar manner.

We prove that $\mathcal{A}$ is a homomorphic image of the cascade $\mathcal{B} \circ \mathcal{C}$. Consider the function

$$\phi \colon Q^{\mathcal{B}\circ\mathcal{C}} \to Q^{\mathcal{A}} \quad \text{defined by} \quad \phi(D_j, C_i) = i + j.$$

The codomain of ϕ is $Q^{\mathcal{A}}$ because if $i = 1, \ldots, n-1$ and $j = 0, 1$ then $i + j \in \{1, \ldots, n\} = Q^{\mathcal{A}}$. Hence, ϕ is surjective. Notice that, as opposed to the Krohn-Rhodes general case, ϕ is a total function.

We prove that ϕ is a homomorphism from $\mathcal{B} \circ \mathcal{C}$ to $\mathcal{A}$, that is, for all $j = 0, 1$ and $i = 1, \ldots, n-1$:

$$\phi\big(\delta^{\mathcal{B}\circ\mathcal{C}}((D_j, C_i), a)\big) = \delta^{\mathcal{A}}\big((\phi(D_j, C_i), a)\big).$$

In fact, we have

$$\phi\big(\delta^{\mathcal{B}\circ\mathcal{C}}((D_j, C_i), a)\big) = \phi\big(\delta^{\mathcal{B}}(D_j, a), \delta^{\mathcal{C}}(C_i, (D_j, a))\big) =$$

$$\phi(D_{\gamma(j,a)}, C_{\delta^{\mathcal{A}}(i+j,a)-\gamma(j,a)}) = \delta^{\mathcal{A}}(i+j,a) - \gamma(j,a) + \gamma(j,a) = \delta^{\mathcal{A}}(\phi(D_j, C_i), a).$$

To conclude the proof of the lemma we prove that the order $C_1 < \cdots < C_{n-1}$ makes $\mathcal{C}$ path coherent. Let $[C_k, C_l]$ be any interval, with $1 \leq k \leq l \leq n-1$. From equation (3.10) it follows that, for all $j \in \{0,1\}$ and for all $a \in \Sigma$,

$$\delta^{\mathcal{C}}\Big([C_k, C_l], (D_j, a)\Big) = \{C_h \ : \ h + \gamma(j,a) \in \delta^{\mathcal{A}}([k+j, l+j], a)\}.$$

By the hypothesis that $\mathcal{A}$ is path coherent it follows that $\delta^{\mathcal{A}}([k+j, l+j], a)$ is an interval, hence also the set $\delta^{\mathcal{C}}\Big([C_k, C_l], (D_j, a)\Big)$ is an interval. □

In the theorem that follows, we will use Lemma 43 to prove one of the main results of this section.

Theorem 44. *KRDTpc Each path coherent semiautomaton $\mathcal{A}$ with $n \geq 2$ states is an homomorphic image of a cascade product*

$$\mathcal{B}_1 \circ \mathcal{B}_2 \circ \mathcal{B}_3 \circ \cdots \circ \mathcal{B}_{n-1}$$

of $n-1$ permutation-reset semiautomata, each one having exactly 2 states.

Proof. As we have already noted in Lemma 43, every semiautomaton with exactly 2 states is a permutation-reset semiautomaton, therefore, if $n = 2$ there is nothing to prove. If $n > 2$, we prove, by induction on k, that for all $1 \leq k \leq n-2$ there are semiautomata $\mathcal{B}_1, \ldots, \mathcal{B}_k, \mathcal{C}_k$ such that each $\mathcal{B}_i$ is a permutation-reset semiautomaton with 2 states, $\mathcal{C}_k$ is a path coherent semiautomaton with $n-k$ states, and $\mathcal{A}$ is an homomorphic image of $\mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k \circ \mathcal{C}_k$.
Base step. If $k = 1$, the thesis follows immediately from Lemma 43.
Inductive step. Assume that the thesis holds for some $1 \leq k < n-2$. Since $\mathcal{C}_k$ is path coherent, we can apply Lemma 43 to find a permutation-reset semiautomaton $\mathcal{B}_{k+1}$ with 2 states and a path coherent semiautomaton $\mathcal{C}_{k+1}$ with $(n-k)-1 = n-(k+1)$ states such that $\mathcal{C}_k$ is an homomorphic image of $\mathcal{B}_{k+1} \circ \mathcal{C}_{k+1}$. As a consequence of statement 2) of Proposition 39, we have that $\mathcal{A}$ is an homomorphic image of

$$(\mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k) \circ (\mathcal{B}_{k+1} \circ \mathcal{C}_{k+1}),$$

hence we can apply statement 1) of Proposition 39 to conclude that

$$(\mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k) \circ (\mathcal{B}_{k+1} \circ \mathcal{C}_{k+1}) = (\mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k \circ \mathcal{B}_{k+1}) \circ \mathcal{C}'_{k+1},$$

where $\mathcal{C}'_{k+1}$ is obtained from $\mathcal{C}_{k+1}$ by renaming the labels of its edges. Since $\mathcal{C}_{k+1}$ is a path coherent semiautomaton with $n-(k+1)$ states, so it is $\mathcal{C}'_{k+1}$, concluding the inductive proof. For $k = n-2$ we obtain that there are semiautomata $\mathcal{B}_1, \ldots, \mathcal{B}_{n-2}, \mathcal{C}_{n-2}$, each one with 2 states, such that $\mathcal{A}$ is an homomorphic image of $\mathcal{B}_1 \circ \cdots \circ \mathcal{B}_{n-2} \circ \mathcal{C}_{n-2}$. Since every semiautomaton with exactly 2 states is a permutation-reset semiautomaton, the cascade

$$\mathcal{B}_1 \circ \cdots \circ \mathcal{B}_{n-2} \circ \mathcal{C}_{n-2}$$

is a cascade product of $n-1$ permutation-reset semiautomata, completing the proof. □

Figure 3.9 shows the cascade decomposition of the path coherent automaton depicted in Figure 3.6.

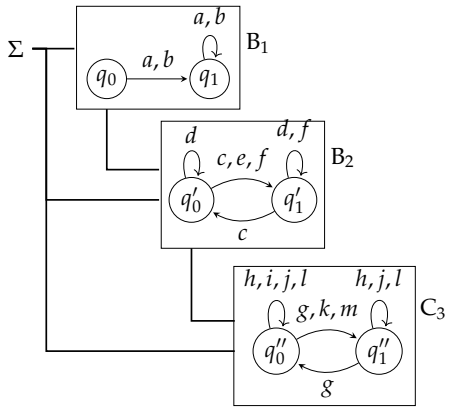


FIGURE 3.9: Cascade decomposition of the automaton in fig. 3.6. The letters $c, d, e, f, g, h, i, j, k, l, m$ are shortcuts for $(q_0, a), (q_0, b), (q_1, a), (q_1, b), (q'_0, c), (q'_0, d), (q'_0, e), (q'_0, f), (q'_1, c), (q'_1, d), (q'_1, e), (q'_1, f)$ respectively.

Note that, although the automaton in Figure 3.6 is counter-free and, as so, KRDT decomposes it using resets only, the decomposition obtained from Theorem 44 (see Figure 3.9) contains blocks with permutations. In this sense, this decomposition is not optimal as the one obtained in the original KRDT. However, as we shall see shortly, in case of path coherence plus (W2) (hence in the Wheeler case as well), in which automata are always counter-free (a consequence of (W2), see proof of Proposition 18), the decomposition that we propose in Theorem 46 allows us to produce a permutation-free cascade.

Lemma 45. *Let $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, \delta^{\mathcal{A}}, <)$ be a path coherent semiautomaton satisfying Wheeler axiom 2 (W2). Then the construction of Lemma 43 provides a reset semiautomaton $\mathcal{B}$ with $|Q^{\mathcal{B}}| = 2$ and a path coherent semiautomaton $\mathcal{C}$ with $|Q^{\mathcal{C}}| = n - 1$ still satisfying (W2) such that $\mathcal{A}$ is an homomorphic image of $\mathcal{B} \circ \mathcal{C}$.*

Proof. Looking at the construction of Lemma 43 we first prove that $\mathcal{B} = \mathcal{A}/\mathcal{D}$ is a reset automaton. Note that $\mathcal{B}$ has only two states. Hence, a transition in $\mathcal{B}$ can only be a (eventually partial) reset, the identity, or the permutation $\delta^{\mathcal{B}}(D_0, a) = D_1$, $\delta^{\mathcal{B}}(D_1, a) = D_0$. Let us prove that the last possibility can not occur if $\mathcal{A}$ is a path coherent semiautomaton satisfying (W2). We reason by way of a contradiction. From the definition of $\delta^{\mathcal{B}}$ given in Lemma 43 we see that if $\delta^{\mathcal{B}}(D_0, a) = D_1$, $\delta^{\mathcal{B}}(D_1, a) = D_0$ then there is an $i \in D_0$ such that $\delta^{\mathcal{A}}(i, a) = n$, while for all $j \in D_1$ it holds $\delta^{\mathcal{A}}(j, a) < n$ — whenever $\delta^{\mathcal{A}}(j, a) \neq \perp$. Hence, $i \notin D_1$ which implies $i = 1$. Since $\delta^{\mathcal{B}}(D_1, a) = D_0$, there is a $j \in D_1$ such that $\delta^{\mathcal{A}}(j, a) \neq \perp$. Since $j \in D_1$ we have $1 < j$, but it also hold $\delta(1, a) = n > \delta(j, a)$, contradicting (W2).

Next we have to prove that states of semiautomaton $\mathcal{C}$, ordered as $C_1 < \cdots < C_{n-1}$, satisfy (W2) (we already proved that it is path coherent in Lemma 43). Consider a $\mathcal{C}$-transition (D_j, a) with $a \in \Sigma$ and $j = 0, 1$

Suppose $C_i < C_{i'}$, that is, $i < i'$. Then we have $i + j < i' + j$ and, from (W2) applied to $\mathcal{A}$, it follows $\delta^{\mathcal{A}}(i + j, a) \leq \delta^{\mathcal{A}}(i' + j, a)$ thus

$$\delta^{\mathcal{A}}(i + j, a) - \gamma(j, a) \leq \delta^{\mathcal{A}}(i' + j, a) - \gamma(j, a),$$

implying

$$\delta^{\mathcal{C}}\Big(C_i, (D_j, a)\Big) = C_{\delta^{\mathcal{A}}(i+j,a)-\gamma(j,a)} \leq C_{\delta^{\mathcal{A}}(i'+j,a)-\gamma(j,a)} = \delta^{\mathcal{C}}\Big(C_{i'}, (D_j, a)\Big).$$

□

Theorem 46. *Each path coherent semiautomaton $\mathcal{A}$ satisfying (W2) with $n \geq 2$ states is the homomorphic image of a cascade product of $n-1$ reset semiautomata, each one having exactly 2 states.*

Proof. The proof is the same as the one of Theorem 44, but we apply Lemma 45 instead of Lemma 43 everywhere. □

Since Wheeler automata are path coherent and satisfy (W2), we get

Corollary 46.1. *Any Wheeler semiautomaton $\mathcal{A}$ with $n \geq 2$ states is the homomorphic image of a cascade product of $n-1$ reset semiautomata, each one having exactly 2 states.*

3.3 Conclusions

In this chapter we proved that path-coherent semiautomata admit a linear-sized cascade decomposition into permutation-reset semiautomata. Moreover we showed that, for the case of Wheeler semiautomata, the cascade is permutation-free. A merit of the provided proofs is their simplicity. What makes them easier to follow compared to the various proofs given in the literature on KRDT is the fact that the order on the set of states imposed respectively by path coherence and Wheelerness allows us to immediately find a admissible decomposition of the states of the initial automaton. This guarantees that the components obtained in the decomposition satisfy condition 2) of Theorem 41, which is the portion of —classic— KRDT difficult to prove.

An interesting open question is the following: is it possible to combine the main result of this chapter, namely Corollary 46.1, with Lemma 20 of Chapter 2, which states that the cascade product of two Wheeler automata is still a Wheeler automaton? If one could prove that each component of the composition obtained by applying Corollary 46.1 is a Wheeler automaton, it would provide a characterization of Wheeler automata as that class of automata decomposable into a cascade of Wheeler reset automata.

Chapter 4

Extending Wheelerness to regular languages

Up until this chapter, we focused our attention on the class of Wheeler automata. Despite their good properties, like compression and indexing, Wheeler automata form a restricted class: a proper subclass of counter-free automata (see Definition 5). In order to extend our analysis to the entire class of finite automata a generalization of Definition 11 is called for. In [23], such a generalization of the Wheeler notion is proposed and analyzed, mostly for deterministic automata. The non deterministic case appears to be less tractable and in the following sections we shall contribute on this part.

4.1 Definitions and previous results

The simplest way to generalize the notion of Wheelerness consist in dropping the totality request. At first glance this may seem to be an oversimplification: axioms (W1′) and (W2′) of Definition 11 are automatically satisfied if we consider the trivial partial order that never compares distinct states. While this is indeed the case, we should consider the fact that many different partial orders may satisfy axioms (W1′) and (W2′), and among them we can try to find the ones "as close as possible" to a total order. The following definition of *width* of a partial order, a measure of how far a given partial order is from being total, will come in handy in the rest of this chapter.

Definition 23. Let $(Z, \leq)$ be a partial order. The *width* of $(Z, \leq)$, denoted by width$(\leq)$, is the maximum cardinality of a subset of $Z' \subseteq Z$ such that each pair of distinct elements $x, y \in Z'$ are incomparable, that is, neither $x \leq y$ nor $y \leq x$.

In particular, a partial order of width 1 is a total order —every pair of elements is comparable. An alternative characterization of the width of a partial order can be given as follows. A *chain-partition* of a partially ordered set is a partition where each subset is totally ordered and Dilworth's Theorem ([42]) states that the width of the partial order coincides with the smallest cardinality of a chain-partition. Hence, in a way, the higher the width of a given partial order, the greater is its "distance" from being total.

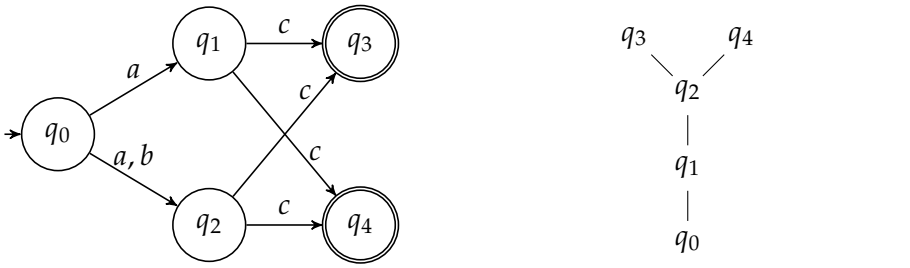
We are going to apply this notion of *width* to a special class of partial orders, called the *colex orders* of a given automaton.

Definition 24. Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA. A *colex order* on $\mathcal{A}$ is a partial order $\leq$ on Q that satisfies the following two axioms:

1. (Axiom 1) For every $u, v \in Q$, if $u < v$, then, if $u \in \delta(u', a)$ and $v \in \delta(v', b)$, it holds $a \preceq b$;

2. (Axiom 2) For every $a \in \Sigma$ and $u, v, u', v' \in Q$, if $u \in \delta(u', a)$, $v \in \delta(v', a)$ and $u < v$, then $u' \leq v'$.

It immediately follows that an automaton has a *colex order* of width 1 —that is, a total colex order— if and only if it is Wheeler. It is then natural to define the *width* of a given automaton $\mathcal{A}$ as the smallest width of a colex order over the states of $\mathcal{A}$. In Figure 4.1, an example of a NFA of width 2 is depicted.



(a) A NFA $\mathcal{A}$.

(b) The Hasse diagram of a colex order of $\mathcal{A}$ of width 2.

FIGURE 4.1: A NFA of width 2. The reader can verify that the order depicted on the right is a colex one, whose width is clearly 2. $\mathcal{A}$ does not admit a colex order of width 1 (that is, a total one): in such hypothetical order, we would have $q_1 < q_2$ due to (Axiom 1). Then, we would not be able to order states q_3, q_4: if $q_3 < q_4$, from $\delta(q_2, c) = q_3$, $\delta(q_1, c) = q_4$ and (Axiom 2) it would follow $q_2 < q_1$, a contradiction. If instead $q_4 < q_3$, a similar contradiction would occur when considering edges $\delta(q_1, c) = q_3$, $\delta(q_2, c) = q_4$.

Definition 25. Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA. The *width* of $\mathcal{A}$, denoted by $\text{width}(\mathcal{A})$, is defined as follows:

$$\text{width}(\mathcal{A}) := \min\{\text{width}(\leq) : \ (Q, \leq) \ \text{ is a colex order}\}.$$

In [23] it is shown that the width of an automaton is a significant parameter to measure the complexity of the automaton: many hard problems become fixed-parameter tractable (FPT) on constant width. The following theorem represents an instance of such results.

Theorem 47. *Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA and let $\mathcal{A}^*$ be the powerset automaton obtained from $\mathcal{A}$, with set of states Q^*. Let $n = |Q|$ and $p = \text{width}(\mathcal{A})$. Then:*

1. *$\text{width}(\mathcal{A}^*) \leq 2^p - 1$;*
2. *$|Q^*| \leq 2^p(n - p + 1) - 1$.*

Next, we consider the notion of width for regular languages.

Definition 26. Let $\mathcal{L}$ be a regular language.

1. The *non-deterministic colex width* of $\mathcal{L}$, denoted by $\text{width}^{nd}(\mathcal{L})$, is the smallest integer p for which there exists an NFA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}$ and $\text{width}(\mathcal{A}) = p$.
2. The *deterministic colex width* of $\mathcal{L}$, denoted by $\text{width}^{d}(\mathcal{L})$, is the smallest integer p for which there exists a DFA $\mathcal{D}$ such that $\mathcal{L}(\mathcal{D}) = \mathcal{L}$ and $\text{width}(\mathcal{D}) = p$.

A notable property that it is missed in the generalization from width 1 —the Wheeler case— to width $p \geq 2$, is the following: the class of languages recognized by Wheeler DFAs coincides with the class of languages recognized by a Wheeler NFAs. In [23] it is noted that this does not generally applies to widths greater than 1, that is, there exist languages where the non-deterministic width is strictly smaller than the deterministic one. Nonetheless, Theorem 47 allows us to give an upper bound to the deterministic width of a language in terms of its non-deterministic width.

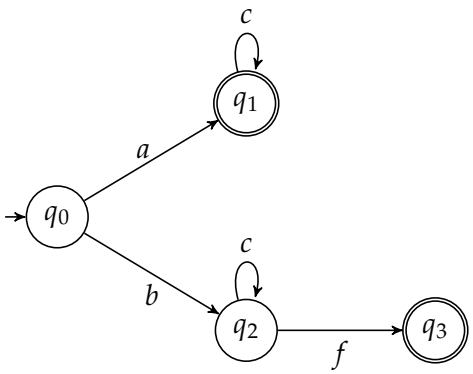
Corollary 47.1. *Let $\mathcal{L}$ be a regular language. Then*

$$width^d(\mathcal{L}) \leq 2^{width^{nd}(\mathcal{L})} - 1.$$

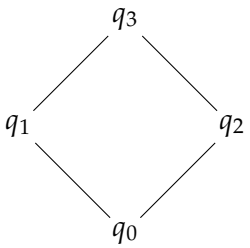
In this chapter we will analyse this aspect in more depth, exploring, in Section **??**, the possible distances between the deterministic and non-deterministic width of regular languages. To do so, we will need to adapt the notion of *entanglement* given in [23]. This measure can be computed on the minimum DFA recognizing a given language $\mathcal{L}$ and uniquely characterizes the deterministic width of $\mathcal{L}$.

Definition 27. Let $\mathcal{D}$ be a DFA with set of states Q. A subset $Q' \subseteq Q$ is *entangled* if there exists a monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$ in $\text{Pref}(\mathcal{L}(\mathcal{D}))$ such that for all $u' \in Q'$ it holds $\delta(s, \alpha_i) = u'$ for infinitely many i's. In this case the sequence $(\alpha_i)_{i\in\mathbb{N}}$ is said to be a *witness* for (the entanglement of) Q'. Moreover, define :

$$\text{ent}(\mathcal{D}) = \max\{|Q'| \mid Q' \subseteq Q \text{ and } Q' \text{ is entangled } \}$$
$$\text{ent}(\mathcal{L}) = \min\{\text{ent}(\mathcal{D}) \mid \mathcal{D} \text{ is a DFA } \wedge\ \mathcal{L}(\mathcal{D}) = \mathcal{L}\}.$$



(a) The minimum DFA $\mathcal{D}_{\mathcal{L}_b}$ recognizing $\mathcal{L}_b = ac^* \cup bc^*f$.



(b) The Hasse diagram of a colex order of $\mathcal{D}_{\mathcal{L}_b}$ whose width is 2.

FIGURE 4.2: A non Wheeler language $\mathcal{L}_b$ whose deterministic width is 2.

Example 2. Consider the minimum DFA $\mathcal{D}_{\mathcal{L}_b}$ of the language $\mathcal{L}_b$ depicted in Figure 4.2. As we have shown in Chapter 2, $\mathcal{L}_b$ is not Wheeler (see Figure 1.5(b)). Since $I_{q_0} = \{\varepsilon\}$ and I_{q_3} contains only strings ending with f, the only two states of $\mathcal{D}_{\mathcal{L}_b}$ that can be entangled are q_1 and q_2. A monotone sequence of strings in $\text{Pref}(\mathcal{L}_b)$ that switches infinitely many times between q_1 and q_2 is the following:

$$a \prec b \prec ac \prec bc \prec \cdots \prec ac^n \prec bc^n \prec \ldots$$

Hence, states q_1, q_2 are entangled and the entanglement of $\mathcal{L}_b$ is 2. In this particular case, the minimum deterministic width is already achieved by the minimum DFA $\mathcal{D}_{\mathcal{L}_b}$, as shown in Figure 4.2(b). Later in this section we will see that this is not always the case (see Example 3).

Notice that we can see the entanglement in another way: if we focus on the prefixes of the language, imagining them arranged on a line from left to right, we observe that there is a point around which infinite prefixes accumulate, as evidenced by the previous inequality. We can observe that actually the sequences accumulating around this point are two: the first composed of prefixes belonging to I_{q_1}, the second composed of prefixes belonging to I_{q_2}. The infinite alternation of these two sequences accumulating around a specific point is the cause of the entanglement between the two states, and in [35] it has been shown that whenever two states are entangled, it is possible to identify a point of accumulation that generates such entanglement.

Notice that if there are k entangled states in the minimum DFA recognizing a language $\mathcal{L}$, splitting them a finite number of times —hence creating a bigger DFA that recognizes $\mathcal{L}$— will never result in a new set of states whose input languages are totally ordered by the relation $\prec$: the same monotone sequence that testifies the entanglement of the original k states also testifies the entanglement of at least k states among the new ones —possibly more. It immediately follows that $\mathrm{ent}(\mathcal{L})$ is a lower bound to the deterministic width of $\mathcal{L}$. The next result proves that the entanglement of the states of the minimum DFA is actually the only obstacle that prevents us from reducing the deterministic width of $\mathcal{L}$.

Theorem 48 ([23]). *If $\mathcal{D}_{\mathcal{L}}$ is the minimum DFA of the regular language $\mathcal{L}$, then:*

$$width^d(\mathcal{L}) = ent(\mathcal{L}) = ent(\mathcal{D}_{\mathcal{L}}).$$

In [23] the previous theorem is used to compute the width of a language in polynomial time.

Concerning the non-deterministic width of a language, our knowledge is far from being complete. In particular, no algorithm is known for computing such width. The absence of a minimum NFA for regular languages makes impossible to follow the same strategy employed in the deterministic case. Moreover, as we will see in the following example[1], in general it is not true that the NFA that fulfills the minimum non-deterministic width of a language $\mathcal{L}$ has less states than $\mathcal{D}_{\mathcal{L}}$: sometimes we need to duplicate some states in order to reduce the width.

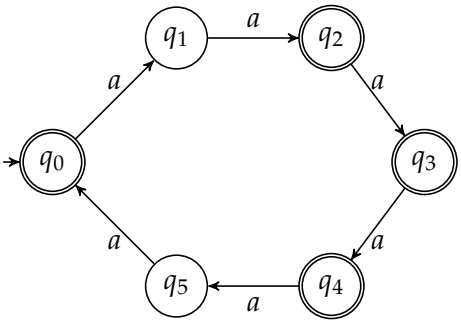
Example 3. Let p, q be two prime numbers and let $\mathcal{L}_{p,q}$ be the language

$$\mathcal{L}_{p,q} := \{a^n \mid p \text{ divides } n \text{ or } q \text{ divides } n\}.$$

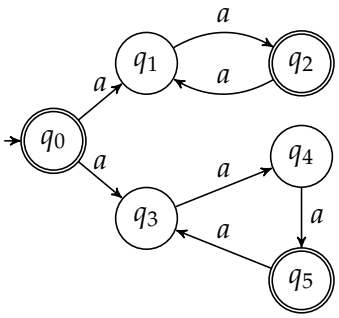
The deterministic width of $\mathcal{L}_{p,q}$, equal to the entanglement of $\mathcal{D}_{\mathcal{L}_{p,q}}$, is pq: the DFA $\mathcal{D}_{\mathcal{L}_{p,q}}$ consists of a simple cycle labeled a^{pq} whose states are all entangled. The non-deterministic width of $\mathcal{L}_{p,q}$ is at most $p+q$: one of the NFAs fulfilling this width has $p+q+1$ states, all but the initial one being entangled, arranged in two simple cycles of length p and q respectively. The case $p = 2$ and $q = 3$ is depicted in Figure 4.3.

Notice that the deterministic width of a regular language $\mathcal{L}$ is an upper bound to the non-deterministic width of $\mathcal{L}$ and a natural question regarding the two widths is:

[1] Provided by Nicola Cotumaccio.



(a) The minimum DFA $\mathcal{D}_{\mathcal{L}_{2,3}}$. Both its entanglement and its width are equal to the number of states, that is, 2*3=6.



(b) A NFA fulfilling the non-deterministic width of $\mathcal{L}_{2,3}$, that is, 2+3=5.

FIGURE 4.3: A DFA and a NFA fulfilling the deterministic and non-deterministic width of $\mathcal{L}_{2,3}$.

is the distance between the two widths fixed, or the gap between them can be made arbitrarily large, up to a certain upperbound?

In Sections 4.2, 4.3 we will generalize the notion of entanglement in order to obtain some lower bounds to the non-deterministic width and answer the previous question.

4.2 On the difference between deterministic and non-deterministic width

As we have seen in the previous section, the entanglement of a language $\mathcal{L}$ — computed on its minimum DFA— is an exact measure of the deterministic width of $\mathcal{L}$. Can we obtain a similar result for the non-deterministic width, possibly by defining an entanglement-like new property over the set of states of the minimun DFA? In this section and the next, we begin a journey towards this result introducing some measures which are approximations of the one we are after, and we will use them to compare the deterministic and non-deterministic width of a regular language.

Recall that, for every regular language $\mathcal{L}$, it holds

$$\text{ent}(\mathcal{L}) = \text{width}^d(\mathcal{L}) \geq \text{width}^{nd}(\mathcal{L}).$$

We start with a simple example where we show that the above inequality may be strict (as already pointed out in [23]).

Example 4. An example of a regular language $\mathcal{L}$ such that $\text{width}^{nd}(\mathcal{L}) = 2$ and $\text{width}^d(\mathcal{L}) = 3$ is the following. Consider the language $\mathcal{L}$ recognized by the automata in Figure 4.4.

Let us prove that the deterministic width of $\mathcal{L}$ is 3. It will suffice to prove that the entanglement of the minimum automaton, on the right in Figure 4.4, is 3. First notice that, from the definition, it follows that entangled states must have an incoming edge labeled by the same letter. Therefore, the entanglement of $\mathcal{D}_{\mathcal{L}}$ is at most 3. Moreover, the entanglement is at least 3, because the monotone sequence

$$\ldots ad^n \prec bd^n \prec cd^n \prec ad^{n+1} \prec \ldots$$

in $Pref(\mathcal{L})$ goes infinitely many times through q_1, q_4, q_5.

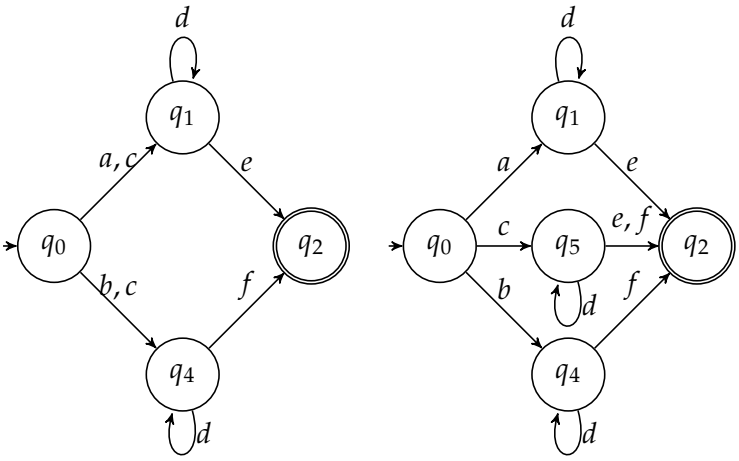


FIGURE 4.4: A non deterministic automaton $\mathcal{A}$ (left) and the minimum DFA $\mathcal{D}_{\mathcal{L}}$ recognizing $\mathcal{L}$ (right).

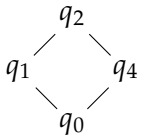


FIGURE 4.5: The Hasse diagram of a colex order $\leq$ for the automaton on the left in Fig. 4.4

Let us prove that the non-deterministic width of $\mathcal{L}$ is 2: it is at least 2, because if it were 1, then the deterministic width would also be equal to 1 (see Corollary 47.1). On the other hand, the colex order $\leq$ of the NFA on the left in Figure 4.4, whose Hasse diagram is depicted in Fig. 4.5, has width equal to 2. Therefore, the non-deterministic width of $\mathcal{L}$ is 2.

We may re-read the above observation as follows: in $\mathcal{D}_{\mathcal{L}}$ we have 3 entangled states q_1, q_4, q_5, that we can identify with their output languages $a^{-1}\mathcal{L}, b^{-1}\mathcal{L}, c^{-1}\mathcal{L}$. However, since

$$c^{-1}\mathcal{L} \subseteq a^{-1}\mathcal{L} \cup b^{-1}\mathcal{L},$$

by giving up determinism we can lower the width by erasing state q_5 and redirecting strings entering q_5 in both q_1 and q_4 (see the NFA $\mathcal{A}$ in Figure 4.4).

This re-reading suggests the following definition:

Definition 28. A set $\{[\alpha_1]_{\mathcal{L}}, [\alpha_2]_{\mathcal{L}}, \ldots, [\alpha_k]_{\mathcal{L}}\}$ of pairwise distinct MN-classes is said to be *relatively indecomposable* if for all i it holds $\alpha_i^{-1}\mathcal{L} \not\subseteq \bigcup_{j\neq i} \alpha_j^{-1}\mathcal{L}$.

In the following, to identify an MN-class $[\alpha]$ of a regular language $\mathcal{L}$, we will use freely both the state q of the minimum automaton reached by α and the language $\alpha^{-1}\mathcal{L}$.

Remember that the entanglement of a language $\mathrm{ent}(\mathcal{L})$ is the minimum number k for which all monotone sequences in the prefixes end up (i.e. go through infinitely many times) in at most k different MN-class. We know that $\mathrm{ent}(\mathcal{L}) = \mathrm{width}^{d}(\mathcal{L})$. To get closer to $\mathrm{width}^{nd}(\mathcal{L})$ we strengthen this condition as follows:

Definition 29. The *entanglement+* of a language, $\text{ent}^+(\mathcal{L})$, is the minimum number k for which all monotone sequences in the prefixes end up (i.e. go through infinitely many times) in at most k different relatively indecomposable classes.

In other words, $\text{ent}^+(\mathcal{L}) = k$ means that there are k relatively indecomposable MN-classes and a monotone sequence going through them infinitely often—that is, they are entangled—, and there are no $k+1$ such classes.

In order to prove that $\text{ent}^+(\mathcal{L})$ is a lower bound on the non-deterministic width of the language we shall use some results regarding the prefix-suffix relation $\preceq_{ps}$ (see Definition 7). First, it can be easily verified that the following lemma holds.

Lemma 49. *If $\mathcal{C}$ is a family of convex sets over a linearly ordered set, then $(\mathcal{C}, \preceq_{ps})$ is a (possibly, non total) order .*

Given an NFA $\mathcal{A} = (Q, s, \delta, F)$ and a colex order $\leq$ on $\mathcal{A}$, we shall use some properties, proved in [23], regarding the sets $I_\alpha = \{u \in Q \mid u \in \delta(q_0, \alpha)\}$, their duals $I_u = \{\alpha \in \text{Pref}(\mathcal{L}(\mathcal{A})) \mid u \in \delta(q_0, \alpha)\}$, and their *ps*-orders induced respectively by $(Q, \leq)$ and $(\text{Pref}(\mathcal{L}(\mathcal{A}), \preceq)$.

Lemma 50 ([23])**.** *Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA and let $\leq$ be a colex order on $\mathcal{A}$. If $\alpha \in Pref(\mathcal{L}(\mathcal{A}))$ then I_α is convex in $(Q, \leq)$.*

Lemma 51 ([23])**.** *Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA and let $\leq$ be a colex order on $\mathcal{A}$. If $u < v$, then $I_u \preceq_{ps} I_v$.*

Actually, we shall use a dual version of the previous lemma, that we prove below.

Corollary 51.1. *Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA, let $\leq$ be a colex order on $\mathcal{A}$ and let $(Q_i)_i$ be a chain partition of $\leq$. If $\alpha \prec \beta$ then $I_\alpha \cap Q_i \leq_{ps} I_\beta \cap Q_i$, for all i. In particular, for all i, the family*

$$\mathcal{C} := \{I_\alpha \cap Q_i : \alpha \in Pref(\mathcal{L}), I_\alpha \cap Q_i \neq \emptyset\}$$

is totally ordered by $\leq_{ps}$.

Proof. Suppose that $\alpha \prec \beta$ and assume, reasoning by contradiction, that $I_\alpha \cap Q_i \not\leq_{ps} I_\beta \cap Q_i$, that is, there exists $u \in I_\alpha \cap Q_i$ and $v \in I_\beta \cap Q_i$ with $\{u, v\} \not\subseteq I_\alpha \cap I_\beta$ such that $v < u$. Then, we have that $\beta \in I_v$, $\alpha \in I_u$ and $\{\alpha, \beta\} \notin I_v \cap I_u$ and we can then apply Lemma 51 to v, u to deduce that $\beta \prec \alpha$, a contradiction. Moreover, from Lemma 50 we know that the set I_α is convex, for all $\alpha \in \text{Pref}(\mathcal{L})$, and so it is $I_\alpha \cap Q_i$, for all α and i. We then apply Lemma 49 to conclude that the family $\mathcal{C}$ is totally ordered. □

When we have a chain partition of a colex order over an NFA, then the set of states reached by any monotone sequence of strings in the prefixes of the accepted language becomes eventually constant in any component of the partition. This formally proved below.

Lemma 52. *Let $\mathcal{A} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a colex order of width p, and let $Q_1, \ldots, Q_p \subseteq Q$ be a chain partition of $(Q, \leq)$. If $(\alpha_i)_{i \in \mathbb{N}}$ is a monotone sequence in $Pref(\mathcal{L})$ then there exists an n such that, for all i and for all $m, m' > n$, if $I_{\alpha_m} \cap Q_i \neq \emptyset$, $I_{\alpha_{m'}} \cap Q_i \neq \emptyset$, then $I_{\alpha_m} \cap Q_i = I_{\alpha_{m'}} \cap Q_i$.*

Proof. For all $i = 1, \ldots, k$ the family

$$\mathcal{C} = \{I_\alpha \cap Q_i : \alpha \in \text{Pref}(\mathcal{L}), I_\alpha \cap Q_i \neq \emptyset\}$$

is a finite family of intervals in Q_i totally ordered by $\leq_{ps}$. Moreover, from Corollary 51.1 it follows that if $\alpha \prec \beta$ and $I_\alpha \cap Q_i \neq \emptyset, I_\beta \cap Q_i \neq \emptyset$ then $I_\alpha \cap Q_i \leq_{ps} I_\beta \cap Q_i$.

Therefore, the sequence of convex sets $(I_{\alpha_m} \cap Q_i)_m$ (restricted to the sets for which $I_{\alpha_m} \cap Q_i \neq \emptyset$) is monotone in the finite order $(\mathcal{C}, \leq_{ps})$. Hence the sequence must be eventually constant. □

We can use the previous lemma to prove that ent^+ is a lower bound to $\text{width}^{nd}(\mathcal{L})$.

Lemma 53. *If $\mathcal{A}$ is an NFA recognizing $\mathcal{L}$ then $ent^+(\mathcal{L}) \leq width(\mathcal{A})$.*

Proof. Let $\text{ent}^+(\mathcal{L}) = k$, $\text{width}(\mathcal{A}) = p$, and let $Q_1, \ldots, Q_p$ be a chain partition of a colex order of width p over $\mathcal{A}$. Suppose, by way of contradiction, that $p < k$, and consider a monotone increasing sequence $(\alpha_i)_{i \in \mathbb{N}}$ (the case of a decreasing sequence is handled similarly) going through k indecomposable MN-classes. Possibly considering a subsequence, we can assume that the previous lemma holds with $n = 0$ and the k-indecomposable classes are $[\alpha_1]_{\mathcal{L}}, \ldots, [\alpha_k]_{\mathcal{L}}$. This implies that $I_{\alpha_1}, \ldots, I_{\alpha_k}$ are pairwise distinct, because $I_{\alpha_i} = I_{\alpha_j}$ implies $\alpha_i^{-1}\mathcal{L} = \alpha_j^{-1}\mathcal{L}$ and so $[\alpha_i]_{\mathcal{L}} = [\alpha_j]_{\mathcal{L}}$, contradicting the indecomposability of the classes $[\alpha_i]_{\mathcal{L}}$.

By Lemma 52, there exist subsets $X_1 \subseteq Q_1, \ldots, X_p \subseteq Q_p$ such that for every i there exists $K_i \subseteq \{1, \ldots, p\}$ with:

$$I_{\alpha_i} \cap Q_j = \begin{cases} X_j & \text{if } j \in K_i \\ \emptyset & \text{otherwise.} \end{cases}$$

So that we have

$$I_{\alpha_i} = \bigcup_{j \in K_i} X_j.$$

Consider the sets $W_i = I_{\alpha_1} \cup I_{\alpha_2} \cup \cdots \cup I_{\alpha_i}$, for $i = 1, \ldots, k$. Clearly $W_1 \subseteq W_2 \subseteq \cdots \subseteq W_k$ and every time the inclusion is strict, the number of considered X_j's increases by at least one. Since $k > p$, one of the inclusions must be an equality. So there exists $i \in \{1, \ldots, k\}$ with

$$I_{\alpha_i} \subseteq \bigcup_{j \neq i} I_{\alpha_j}.$$

However, the previous inclusion implies that

$$\alpha_i^{-1}\mathcal{L} \subseteq \bigcup_{j \neq i} \alpha_j^{-1}\mathcal{L},$$

contradicting the indecomposability hypothesis: if $\beta \in \alpha_i^{-1}\mathcal{L}$, then $\alpha_i\beta \in \mathcal{L}$. Hence there must exists $u \in I_{\alpha_i}$ such that $\delta(u, \beta) \in \mathcal{L}$. From $I_{\alpha_i} \subseteq \bigcup_{j \neq i} I_{\alpha_j}$ we obtain that there exists $j \neq i$ such that $u \in I_{\alpha_j}$, hence $\alpha_j\beta \in \mathcal{L}$ and $\beta \in \alpha_j^{-1}\mathcal{L}$. □

Corollary 53.1. *If $\mathcal{L}$ is a regular language, then $ent^+(\mathcal{L}) \leq width^{nd}(\mathcal{L})$.*

Moreover, we also have the following.

Corollary 53.2. *If $\mathcal{L}$ is a regular language with $ent^+(\mathcal{L}) = ent(\mathcal{L})$ then $width^{nd}(\mathcal{L}) = width^{d}(\mathcal{L}) = ent^+(\mathcal{L}) = ent(\mathcal{L})$.*

Proof. By the previous corollary, if $\mathcal{L}$ is a regular language then

$$\text{ent}^+(\mathcal{L}) \leq \text{width}^{nd}(\mathcal{L}) \leq \text{width}^{d}(\mathcal{L}) = \text{ent}(\mathcal{L})$$

and the conclusion follows. □

Unfortunately, in general it is not true that $\text{ent}^+(\mathcal{L}) = \text{width}^{nd}(\mathcal{L})$ as we shall see in the following example.

Example 5. We modify the DFA in Fig. 4.4 by adding the transition (q_5, q_2, g), see Figure 4.6. Notice that the resulting DFA is still minimum and the only entangled

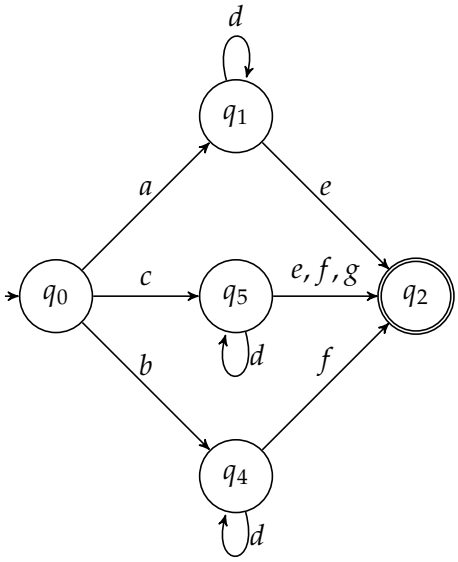


FIGURE 4.6: The minimun automaton of a language such that $\text{ent}^+(\mathcal{L}) \neq \text{width}^{nd}(\mathcal{L})$.

states are q_1, q_4, q_5, which immediately implies $\text{ent}^+(\mathcal{L}) \leq 3$. The MN-classes of the three entangled states are not relatively indecomposable: for instance,

$$d^*e = a^{-1}\mathcal{L} \subseteq b^{-1}\mathcal{L} \cup c^{-1}\mathcal{L} = d^* \cdot \{e, f, g\}.$$

Therefore $\text{ent}^+(\mathcal{L}) \neq 3$ and it is easy to verify that the classes $a^{-1}\mathcal{L}$ and $b^{-1}\mathcal{L}$ are relatively indecomposable, hence $\text{ent}^+(\mathcal{L}) = 2$.We now prove that $\text{width}^{nd}(\mathcal{L}) \geq 3$. Suppose there exists an NFA $\mathcal{A}$ with set of states Q such that $\text{width}(\mathcal{A}) = 2$ and let Q_1, Q_2 be a chain partition of the set of $\mathcal{A}$ states. Consider the monotone sequence

$$\ldots ad^n \prec bd^n \prec cd^n \prec ad^{n+1} \prec bd^{n+1} \prec cd^{n+1} \prec \ldots.$$

Fix n. The sets $I_{ad^n}, I_{bd^n}, I_{cd^n} \subseteq Q$ must be pairwise distinct, since the strings ad^n, bd^n, cd^n belongs to different MN-classes. By Lemma 52, for a sufficiently large n, one among $I_{ad^n}, I_{bd^n}, I_{cd^n}$ is the union of the other two sets, and this in turn implies that one among the MN-classes $(ad^n)^{-1}\mathcal{L}$, $(bd^n)^{-1}\mathcal{L}$, $(cd^n)^{-1}\mathcal{L}$ is the union of the other two, which is not (as can be easily checked on the minimum DFA).

Despite the fact that $\text{ent}^+(\mathcal{L})$ and $\text{width}^{nd}(\mathcal{L})$, in general, do not coincide, this measure is still useful, as we shall see, to construct meaningful examples. We start analyzing, for a given language, the possible distances between its deterministic and non-deterministic widths, showing that for a simple class of automata the two coincide.

Example 6. For any $n \geq 1$ we describe a language $\mathcal{C}_n$ over the alphabet $\{a\}$ such that $\text{width}^{nd}(\mathcal{C}_n) = \text{width}^{d}(\mathcal{C}_n) = n$. We present the language $\mathcal{C}_n$ via its minimum DFA D_n, which is an a-cycle:

- the set of states is $\{q_0, \ldots, q_{n-1}\}$;
- q_0 is initial and final;
- $\delta(q_i, a) = q_{i+1}$, for $0 \leq i < n-1$, $\delta(q_{n-1}, a) = q_0$.

In other words, we have $\mathcal{C}_n = \{a^{nk} \mid k \geq 0\}$. Notice that $\text{ent}^+(\mathcal{L}_n) = \text{ent}(\mathcal{L}_n) = n$ (because there are n Myhill-Nerode classes, they are indecomposable, and the sequence $a, aa, aaa \ldots$ goes trough all of them), so that $\text{width}^{nd}(\mathcal{L}_n) = \text{width}^{d}(\mathcal{L}_n) = n$.

The following lemma considers the other direction, finding, for any n, a language $\mathcal{L}_n$ such that the distance between the deterministic width and the non-deterministic width of the language reaches its maximum —see Corollary 47.1.

Lemma 54. *There exists a family of languages $(\mathcal{S}_n)_{n\geq 1}$ such that $\text{width}^{nd}(\mathcal{S}_n) = n$ and $\text{width}^{d}(\mathcal{S}_n) = 2^n - 1$.*

Proof. Fix $n \geq 1$ and let Σ_n be the alphabet

$$\Sigma_n = \{a_S : \emptyset \subsetneq S \subseteq \{1, \ldots, n\}\} \cup \{x\}.$$

Let $\prec$ be any order over Σ_n that satisfies the following properties, where we use the notation $a_{i_1,\ldots,i_k}$ to denote $a_{\{i_1,\ldots,i_k\}}$:

1. $a_1 \prec a_2 \prec \cdots \prec a_n$,
2. $x \prec a_S \prec a_1$ for all $S \subseteq \{1, \ldots, n\}$ such that $|S| \geq 2$.

The order among characters $a_S, a_{S'}$ such that $|S|, |S'| \geq 2$ is irrelevant for our argument in this example. We will construct a language $\mathcal{S}_n$ over Σ_n such that $\text{width}^{nd}(\mathcal{S}_n) = n$ and $\text{width}^{d}(\mathcal{S}_n) = 2^n - 1$. We present the language $\mathcal{S}_n$ via its minimum DFA B_n (see Fig. 4.7):

- the set of states is $\{q_0\} \cup \{q_f\} \cup \{q_S : \emptyset \subsetneq S \subseteq \{1, \ldots, n\}\}$;
- q_0 is the initial state and q_f the final state;
- $\delta(q_0, a_S) = q_S$, for each $\emptyset \subsetneq S \subseteq \{1, \ldots, n\}$;
- $\delta(q_S, x) = q_S$, for each $\emptyset \subsetneq S \subseteq \{1, \ldots, n\}$;
- $\delta(q_S, a_i) = q_f$, for each $\emptyset \subsetneq S \subseteq \{1, \ldots, n\}$ and $i \in S$.

It is immediate to check that this is actually a minimum DFA. Moreover, notice that $\text{ent}(B_n) = 2^n - 1$. For simplicity we prove this in the case $n = 3$ (see Figure 4.7), but the proof can easily be adapted to any n. Assuming that $x \prec a_{1,2,3} \prec a_{2,3} \prec a_{1,3}, \prec a_{1,2} \prec a_1 \prec a_2 \prec a_3$, we have that the following monotone sequence goes through $2^n - 1$ states of B_n, showing that $\text{ent}(B_n) \geq 2^n - 1$: $a_3 \succ a_2 \succ a_1 \succ a_{1,2} \succ a_{1,3} \succ a_{2,3} \succ a_{1,2,3} \succ a_3x \succ a_2x \succ a_1x \succ a_{1,2}x \succ a_{1,3}x \succ a_{2,3}x \succ a_{1,2,3}x \succ a_3x^2 \succ \ldots$ Moreover, since entangled states must have an input letter in common, we have $\text{ent}(B_n) \leq 2^n - 1$. From $\text{ent}(B_n) = 2^n - 1$ and Theorem 48 we obtain $\text{width}^{d}(\mathcal{S}_n) = 2^n - 1$.

At the same time, we have $\text{ent}^+(\mathcal{S}_n) \geq n$, since the sequence $a_n \succ a_{n-1} \succ \cdots \succ a_1 \succ a_nx \succ a_{n-1}x \succ \ldots a_1x \succ a_3x^2 \succ \ldots$ is a monotone sequence going through n indecomposable Myhill-Nerode classes (see Fig.4.7). Using Lemma 53 we obtain $\text{width}^{nd}(\mathcal{S}_n) \geq n$.

Hence, to prove that $\text{width}^{nd}(\mathcal{S}_n) = n$ we only have to describe an NFA $\mathcal{N}_n$ such that $\text{width}(\mathcal{N}_n) = n$. $\mathcal{N}_n$ can be defined as follows (see Figure 4.8):

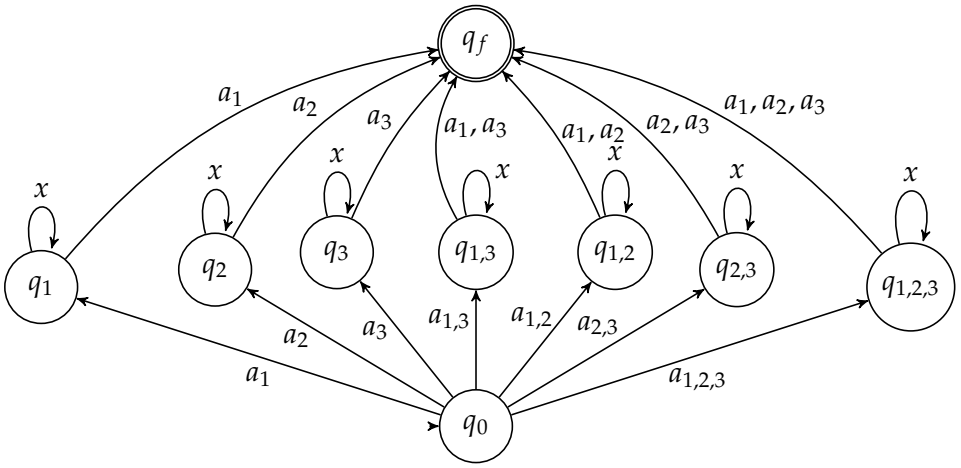


FIGURE 4.7: The minimum DFA B_3 for $\mathcal{S}_3$

- the set of states is $\{q_0\} \cup \{q_f\} \cup \{q_i : i \in \{1, \ldots, n\}\}$;
- q_0 is the initial state and q_f the final state;
- $\delta(q_0, a_S) = \{q_i : i \in S\}$, for each $\emptyset \subsetneq S \subseteq \{1, \ldots, n\}$;
- $\delta(q_i, x) = \{q_i\}$, for each $i \in \{1, \ldots, n\}$;
- $\delta(q_i, a_i) = \{q_f\}$, for each $i \in \{1, \ldots, n\}$.

It is immediate to check that $\mathcal{N}_n$ recognizes $\mathcal{S}_n$. Consider the partial order such that $q_0 < q_1 < q_f$, with all remaining states pairwise incomparable. This is clearly a colex order of width n.

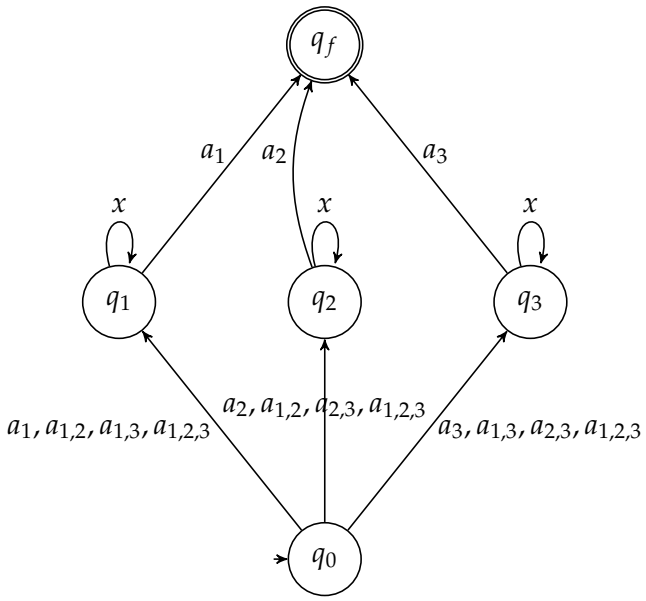


FIGURE 4.8: The NFA $\mathcal{N}_3$ recognizing $\mathcal{L}_3$

□

Notice that in Lemma 54 the alphabet Σ_n of the language $\mathcal{S}_n$ depends on n. In the next lemma we will discharge such dependency by providing a new family of languages $(\mathcal{L}_n)_{n \geq 1}$ over the fixed alphabet $\{a, b\}$.

Lemma 55. *There exists a family of languages $(\mathcal{L}_n)_{n\geq 1}$ over the alphabet $\{a,b\}$ such that $\mathrm{width}^{nd}(\mathcal{L}_n) = n$ and $\mathrm{width}^{d}(\mathcal{L}_n) = 2^n - 1$.*

Proof. Let A_n be the set $A_n = \{0 \ldots, n-1\}$ and let $f : 2^{A_n} \to \{0, \ldots, 2^n - 1\}$ be the bijection defined, for each $S \subseteq A_n$, as

$$f(S) = \sum_{i \in S} 2^i.$$

We present the language $\mathcal{L}_n$ via the following NFA $\mathcal{A}_n$, as (partially) depicted in Figure 4.9. See also the explanation provided in the caption.

- the set of states is $\{q_1, \ldots, q_{2^n-1}\} \cup \{p_0, \ldots, p_{n-1}\} \cup \{r_0, \ldots, r_{n-1}\}$;
- q_1 is the initial state and r_0 is the final state;
- $\delta(q_i, a) = \{q_{i+1}\}$, for $1 \leq i \leq 2^n - 2$;
- $\delta(q_i, b) = \{p_j : \ j \in f^{-1}(i)\}$, for $1 \leq i \leq 2^n - 1$;
- $\delta(p_i, b) = p_i$ and $\delta(p_i, a) = r_i$, for $0 \leq i \leq n-1$;
- $\delta(r_i, a) = r_{i-1}$, for $n - 1 \geq i \geq 1$.

From Lemma 51 it follows that states $p_0, \ldots, p_{n-1}$ are incomparable w.r.t. any colex order on $\mathcal{A}_n$, since so are the sets $I_{p_0}, \ldots, I_{p_{n-1}}$ with respect to the relation $\preceq_{ps}$: for all $0 \leq i < j \leq n-1$ we have that

$$a^{2^i-1}b \preceq a^{2^j-1}b \preceq a^{2^i-1}bb,$$

where

$$a^{2^i-1}b,\ a^{2^i-1}bb \in I_{p_i} \setminus I_{p_j} \text{ and } a^{2^j-1}b \in I_{p_j} \setminus I_{p_i}.$$

Consider the partial order $<$ on the states such that $r_i < p_i$ for all $0 \leq i \leq n-2$ and $q_1 < \cdots < q_{2^n-1} < r_{n-1} < p_{n-1}$ holds. Since it is easily verified that this is a colex order of width equal to n, we obtain $\mathrm{width}(\mathcal{A}_n) = n$, implying $\mathrm{width}^{nd}(\mathcal{L}_n) \leq n$ and

$$\mathrm{width}^{d}(\mathcal{L}_n) \leq 2^{\mathrm{width}^{nd}(\mathcal{L}_n)} - 1 \leq 2^n - 1.$$

Consider now the following DFA $\mathcal{D}_n$ recognizing $\mathcal{L}_n$ and depicted in Figure 4.10:

- the set of states is $\{q_1, \ldots, q_{2^n-1}\} \cup \{p_1, \ldots, p_{2^n-1}\} \cup R$, where the set R will be specified later;
- q_1 is the initial state and all final states belong to R;
- $\delta(q_i, a) = \{q_{i+1}\}$, for $1 \leq i \leq 2^n - 2$;
- $\delta(q_i, b) = \delta(p_i, b) = p_i$, for $1 \leq i \leq 2^n - 1$;
- to each state p_i, for $1 \leq i \leq 2^n - 1$, we append a simple path with all edges labeled a of length $m_i = \max\left[f^{-1}(i)\right]$. That is, we add m_i states $r_{i,0}, \ldots, r_{i,m_i}$, the edges $(p_i, a, r_{i,0})$ and the edges $(r_{i,j}, a, r_{i,j+1})$, for $1 \leq i \leq 2^{n-1} - 1$ and $0 \leq j < m_i$;
- we mark as final the states $r_{i,j}$ such that $j \in f^{-1}(i)$.

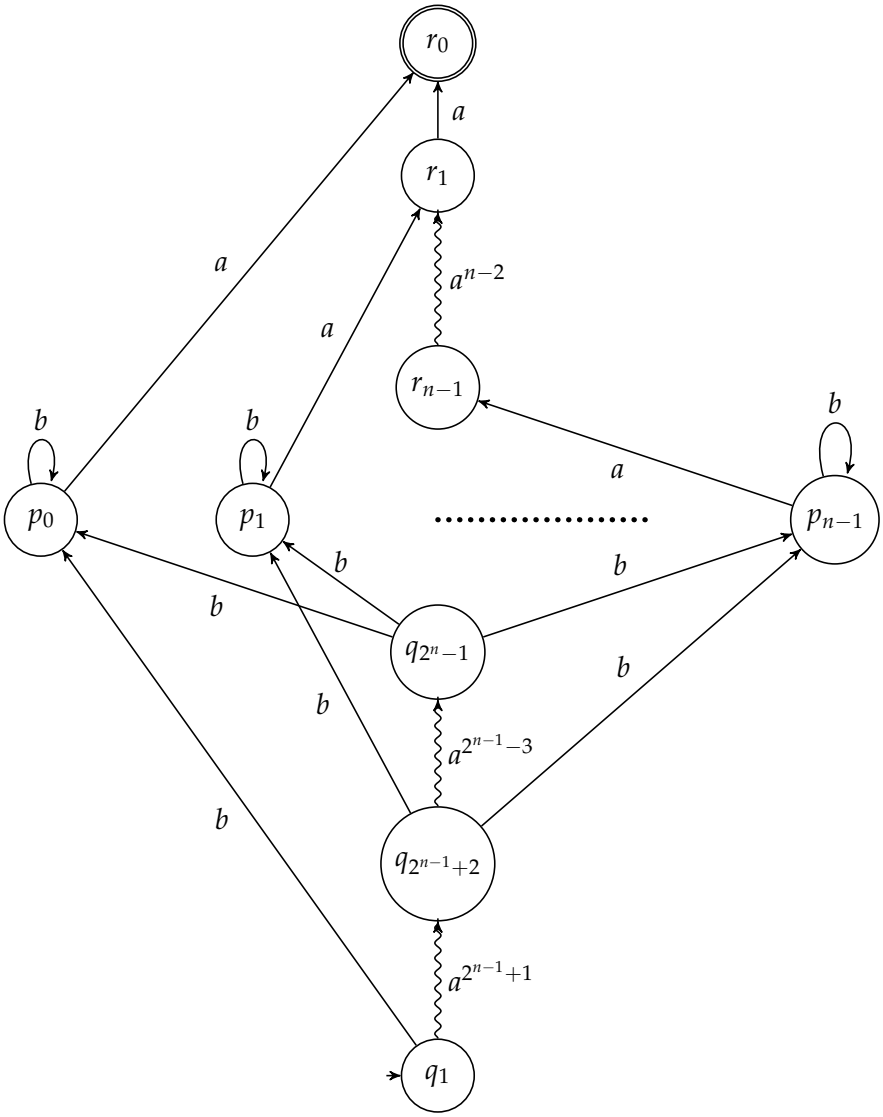


FIGURE 4.9: Partial representation of the NFA $\mathcal{A}_n$. Wavy edges labeled a^m, with $m \in \mathbb{N}$, represent a simple path of length m with all edges labeled a. State $q_{2^{n-1}+2}$ reaches p_1 and p_{n-1} since $f^{-1}(2^{n-1} + 2) = \{1, n-1\}$.

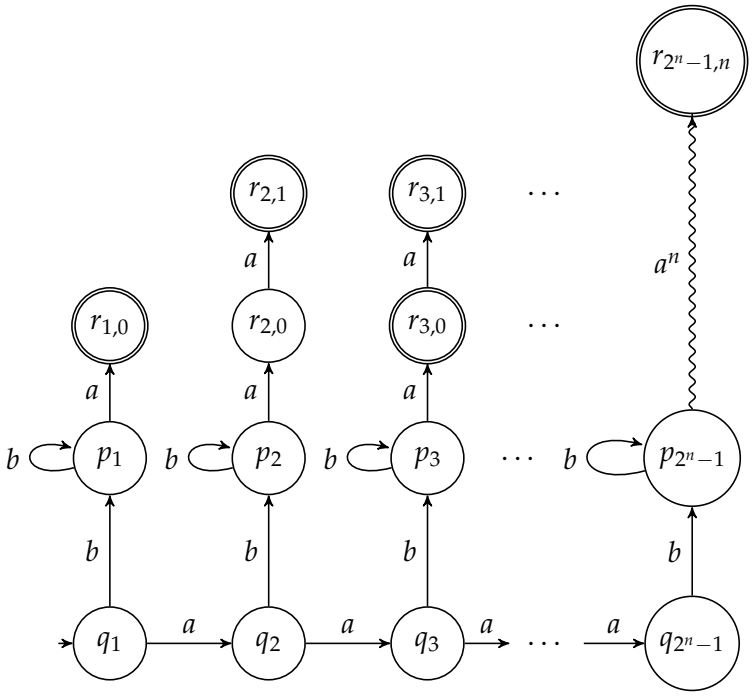


FIGURE 4.10: The DFA $\mathcal{D}_n$. Edges labeled d^x, with $d \in \Sigma$ and $x \in \mathbb{N}$, represent a simple path of length x with all edges labeled d. States $r_{3,0}, r_{3,1}$ are final since $f^{-1}(3) = \{0, 1\}$.

Clearly, states $p_1, ..., p_{2^n-1}$ are entangled. Consider the minimum DFA $\mathcal{D}_{\mathcal{L}_n}$ that recognizes $\mathcal{L}_n$: it can be obtained from $\mathcal{D}_n$ by applying Hoprcroft's algorithm [36], i.e. collapsing states with the same output language. Distinct states p_i, p_j can not be collapsed: let x be an integer in the symmetric difference between $f^{-1}(i)$ and $f^{-1}(j)$, then a^x belongs to either O_{p_i} or O_{p_j}, but not both (see Definition 2). In the minimum DFA $\mathcal{D}_{\mathcal{L}_n}$ there must exist $2^n - 1$ states $p'_1, ..., p'_{2^n-1}$ such that $O_{p'_i} = O_{p_i}$, obtained by collapsing states of $\mathcal{D}_n$ with the same output languages. The input language $I_{p'_i}$ is the union of the input languages of the states of $\mathcal{D}_n$ with the same output language of p_i, hence we have $I_{p'_i} \supseteq I_{p_i}$ for all $0 \leq i \leq n-1$. This implies that also the states $p'_1, ..., p'_{2^n-1}$ are entangled, thus $\text{width}^d(\mathcal{L}_n) \geq 2^n - 1$. Since we had already established that $\text{width}^d(\mathcal{L}_n) \leq 2^n - 1$, the equality must hold; since we already proved that $\text{width}^{nd}(\mathcal{L}_n) \leq n$, the equality $\text{width}^d(\mathcal{L}_n) = 2^n - 1$ in turn implies that $\text{width}^{nd}(\mathcal{L}_n) = n$. □

We can easily modify Lemma 55 to obtain, for any n and any $1 \leq k \leq 2^n - 1 - n$, a language $\mathcal{L}_{n,k}$ such that $\text{width}^{nd}(\mathcal{L}_{n,k}) = \text{ent}^+(\mathcal{L}_{n,k}) = n$ and $\text{width}^d(\mathcal{L}_{n,k}) = ent(\mathcal{L}_{n,k}) = n + k$. This result, proved in the following corollary, shows that every possible distance between the deterministic and the non-deterministic width of a language is achievable.

Corollary 55.1. *For any $n \geq 1$ and any $1 \leq k \leq 2^n - 1 - n$, there exists a language $\mathcal{L}_{n,k}$ such that $\text{width}^{nd}(\mathcal{L}_{n,k}) = ent^+(\mathcal{L}_{n,k}) = n$ and $\text{width}^d(\mathcal{L}_{n,k}) = ent(\mathcal{L}_{n,k}) = n + k$.*

Proof. Fix an $n \geq 1$ and an integer $1 \leq k \leq 2^n - 1 - n$, and consider the automaton $\mathcal{D}_n$ in Figure 4.10. Let I be any subset of $\{1, \ldots, 2^n - 1\}$ of cardinality $n + k$ such that $2^i \in I$ for all $0 \leq i \leq n-1$ and $2^n - 1 \in I$, and let $\mathcal{D}_{n,k}$ be the DFA obtained from

$\mathcal{D}_n$ by removing all the states p_i such that $i \notin I$. We assume that $\mathcal{D}_{n,k}$ is trimmed, that is, we also remove every states $r_{i,j}$ such that $i \notin I$, and we denote by $\mathcal{L}_{n,k}$ the language recognized by $\mathcal{D}_{n,k}$. Note that from $2^n - 1 \in I$ it follows that p_{2^n-1} has not been deleted, thus none of the states q_i can be trimmed, for $1 \leq i \leq 2^n - 1$. Recall that states $p_1, ..., p_{2^n-1}$ were entangled in $\mathcal{D}_n$; similarly, states $\{p_i : \ i \in I\}$ are entangled in $\mathcal{D}_{n,k}$, since they retain the same input languages. As it was for the language $\mathcal{L}_n$ (see Corollary 55), in $\mathcal{D}_{\mathcal{L}_{n,k}}$ —the minimum DFA recognizing $\mathcal{L}_{n,k}$ — there must exist $n + k$ states $\{p'_i : \ i \in I\}$ such that $O_{p'_i} = O_{p_i}$ and $I_{p'_i} \supseteq I_{p_i}$, for all $i \in I$. Thus $\{p'_i : \ i \in I\}$ are entangled, which implies $\text{width}^d(\mathcal{L}_{n,k}) \geq n + k$. We prove now that $\text{width}(\mathcal{D}_{n,k}) = n + k$, thus proving that $\text{width}^d(\mathcal{L}_{n,k}) = n + k$, by providing $n + k$ chains of states of $\mathcal{D}_{n,k}$ in its maximum colex order $\leq$. The first chain is $q_1 < \cdots < q_{2^n-1} < r_{1,0} < p_1$, since $I_{q_1} \prec_{ps} \cdots \prec_{ps} I_{p_1}$. The remaining $n + k - 1$ chains are $r_{i,m_i} < \cdots < r_{i,0} < p_i$ for all $i \in I \setminus \{1\}$.

To prove that $\text{width}^{nd}(\mathcal{L}_{n,k}) = n$, first we apply Lemma 53 to obtain n as a lower bound for the non-deterministic width of $\mathcal{L}_{n,k}$, and then we show a NFA $\mathcal{A}_{n,k}$ that realizes such width. Consider the set of states $P_{pow} = \{p_{2^i} : \ 0 \leq i \leq n-1\}$; this is indeed a subset of states of $\mathcal{D}_{n,k}$, since by hypothesis it holds $2^i \in I$ for all $0 \leq i \leq n-1$. Since $f^{-1}(2^i) = \{i\}$, by construction we have $O_{p_{2^i}} = b^*a^i$, therefore states in P_{pow} are relatively indecomposable. As usual, there must exist n states $\{p'_i : \ 0 \leq i \leq n-1\}$ in $\mathcal{D}_{\mathcal{L}_{n,k}}$ such that $O_{p'_i} = O_{p_{2^i}}$ and $I_{p'_i} \supseteq I_{p_{2^i}}$, for all $0 \leq i \leq n-1$. These states are relatively indecomposable, thus $\text{ent}^+(\mathcal{L}_{n,k}) \geq n$ (see Definition 29) and from Lemma 53 it follows that $n \leq \text{ent}^+(\mathcal{L}_{n,k}) \leq \text{width}^{nd}(\mathcal{L}_{n,k})$. To conclude the proof, we show an NFA $\mathcal{A}_{n,k}$ of width n that recognizes $\mathcal{L}_{n,k}$. Consider the NFA $\mathcal{A}_n$ in Figure 4.9: delete all the edges labeled b that leave any state q_i such that $i \notin I$. Note that this automaton is already trimmed: the edges connecting state q_{2^n-1} to the each of the states $p_0, \ldots, p_{n-1}$ were not deleted, therefore each state of $\mathcal{A}_{n,k}$ is reachable and can reach the final state r_0. Clearly, states $p_0, \ldots, p_{n-1}$ are still incomparable and the n chains of $\mathcal{A}_n$ are still chains of $\mathcal{A}_{n,k}$. Thus $\text{width}(\mathcal{A}_{n,k}) = n$, concluding the proof. □

4.3 A better lower bound to the non-deterministic width

As we proved in the previous section, ent^+ is a lower bound to the non-deterministic width of a language but, as shown in Figure 4.6, this bound is not tight. We can improve this lower bound by modifying the conditions given in Definition 28.

Definition 30. A set $\{[\alpha_1]_{\mathcal{L}}, [\alpha_2]_{\mathcal{L}}, \ldots, [\alpha_k]_{\mathcal{L}}\}$ of pairwise distinct MN-classes is said to be *uniquely decomposable* if for all $K, K' \subseteq \{1, \ldots, k\}$ with $K \cap K' = \emptyset$ it holds

$$\bigcup_{i \in K} \alpha_i^{-1}\mathcal{L} \neq \bigcup_{j \in K'} \alpha_j^{-1}\mathcal{L}.$$

Definition 31. The entanglement* of a language, $\text{ent}^*(\mathcal{L})$, is the maximum number k such that there exist k entangled, uniquely decomposable MN-calsses.

In other words, if $\text{ent}^*(\mathcal{L}) = k$ then there are k uniquely decomposable MN-classes and a monotone sequence going through them infinitely often, but there are no $k + 1$ such classes.

It should be clear from their definitions that, given a regular language $\mathcal{L}$, it holds that:

$$\text{ent}^+(\mathcal{L}) \leq \text{ent}^*(\mathcal{L}).$$

We will prove that ent* is still a lower bound to width$^{nd}(\mathcal{L})$ using the following combinatorial result[2].

Lemma 56. *Let $A = \{x_1, \ldots, x_n\}$ be a set with $|A| = n$ and let $F = \{A_1, \ldots, A_m\}$ be a subset of $Pow(A) \setminus \{\emptyset\}$. Then, $m > n$ implies the existence of two nonempty sets $K, K' \subseteq \{1, \ldots, m\}$ such that $K \cap K' = \emptyset$ and*

$$\bigcup_{i \in K} A_i = \bigcup_{j \in K'} A_j$$

Proof. Let $\vec{a}_i \in \{0,1\}^n$ be the characteristic vector of A_i, for every $i = 1, \ldots, m$, and let $M = (\vec{a}_1, \ldots, \vec{a}_m) \in \{0,1\}^{n \times m}$. Since $m > n$, the $\mathbb{R}^n$-vectors $\vec{a}_1, \ldots, \vec{a}_m$ are linearly dependent. Thus, there exists $\vec{\lambda} = (\lambda_i)_{i=1,\ldots,m} \neq \vec{0}$ with $\lambda_i \in \mathbb{R}$ such that $M \cdot \vec{\lambda} = \vec{0}$. Define $K = \{i \mid \lambda_i > 0\}$ and $K' = \{i \mid \lambda_i < 0\}$. Since every $\vec{a}_i$ is a 0/1-vector, both K and K' are non-empty and disjoint by construction. Finally:

$$\begin{aligned}
x \in \bigcup_{i \in K} A_i &\iff (\exists i \in K)(x \in A_i) \\
&\iff (\exists i = 1, \ldots, m)(\lambda_i > 0 \wedge \vec{a}_{i,x} = 1) \\
&\iff (\exists j = 1, \ldots, m)(\lambda_j < 0 \wedge \vec{a}_{j,x} = 1) \\
&\iff (\exists j \in K')(x \in A_j) \\
&\iff x \in \bigcup_{j \in K'} A_j.
\end{aligned}$$

□

Corollary 56.1. *If $\mathcal{A}$ is an NFA recognizing $\mathcal{L}$ then ent$^*(\mathcal{L}) \leq$ width$(\mathcal{A})$.*

Proof. Let ent$^*(\mathcal{L}) = k$, width$(\mathcal{A}) = p$, and let $Q_1, \ldots, Q_p$ be a chain partition of a colex order of width p over $\mathcal{A}$. Suppose, by way of contradiction, that $p < k$. Using the same argument given in the proof of Lemma 53, we can prove both the existence of a monotone increasing sequence $(\alpha_i)_{i \in \mathbb{N}}$ going through k indecomposable MN-classes $[\alpha_1]_{\mathcal{L}}, \ldots, [\alpha_k]_{\mathcal{L}}$ and the existence of subsets $X_1 \subseteq Q_1, \ldots, X_p \subseteq Q_p$ such that, for every $i = 1, \ldots, k$, there exists $K_i \subseteq \{1, \ldots, p\}$ with:

$$I_{\alpha_i} = \bigcup_{j \in K_i} X_j.$$

We can apply Lemma 56 to the sets $A := \{X_1, \ldots, X_p\}$ and

$$F := \{\bar{I}_{\alpha_1}, \ldots, \bar{I}_{\alpha_k}\} \quad \text{with} \quad \bar{I}_{\alpha_i} := \{X_j : j \in K_i\}$$

to derive the existence of $K, K' \subseteq \{1, \ldots, k\}$ such that $K \cap K' = \emptyset$ and

$$\bigcup_{i \in K} \bar{I}_{\alpha_i} = \bigcup_{j \in K'} \bar{I}_{\alpha_j}$$

Since, given a string α, the set $\alpha^{-1}\mathcal{L}$ only depends on the set of states I_α, this equality implies that

$$\bigcup_{i \in K} \alpha_i^{-1}\mathcal{L} = \bigcup_{j \in K'} \alpha_j^{-1}\mathcal{L},$$

contradicting the unique decomposability hypothesis (see Definition 30). □

[2]The elegant proof below was provided by Brian Riccardi.

In general it is not true that $\text{ent}^*(\mathcal{L}) = \text{width}^{nd}(\mathcal{L})$: let $\mathcal{L}$ be the language recognized by the DFA shown in Figure 4.11.

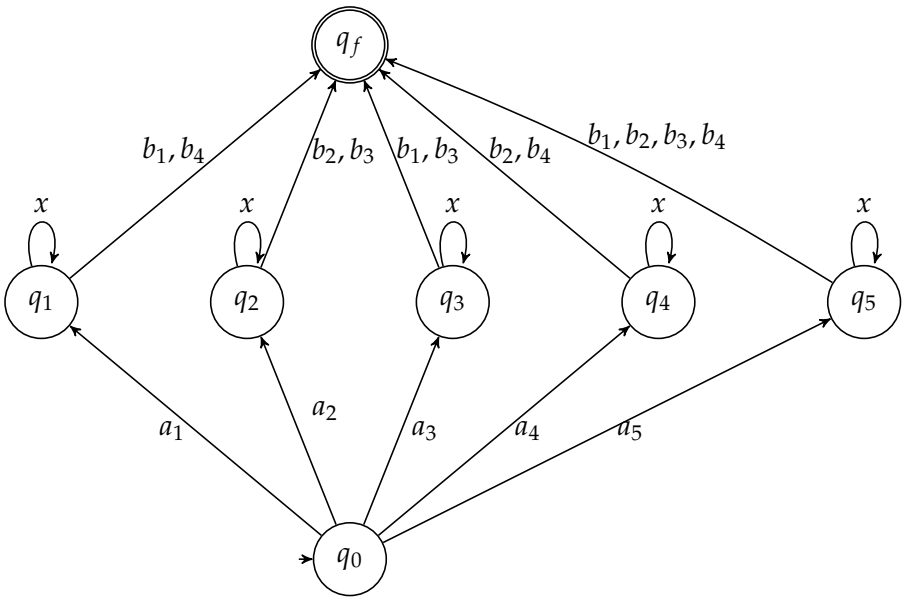


FIGURE 4.11: The minimum DFA recognizing $\mathcal{L}$. We have $\text{ent}^*(\mathcal{L}) = 3$.

Clearly, $\text{ent}^*(\mathcal{L})$ is at least 3, since q_1, q_2, q_3 are entangled and their corresponding Myhill-Nerode classes are uniquely decomposable. One can easily check that it is not possible to choose 4 Myhill-Nerode classes (among the ones corresponding to $q_1, \ldots, q_5$) which are uniquely decomposable. For instance, the classes corresponding to q_1, q_2, q_3, q_4 are not uniquely decomposable since

$$a_1^{-1}\mathcal{L} \cup a_2^{-1}\mathcal{L} = a_3^{-1}\mathcal{L} \cup a_4^{-1}\mathcal{L} = x^* \cdot \{b_1, b_2, b_3, b_4\}.$$

Therefore the entanglement* of $\mathcal{L}$ must be 3.

In Figure 4.12 it is shown an NFA recognizing $\mathcal{L}$ of width 4. This is indeed the non-deterministic width of $\mathcal{L}$, because we can prove that if $\mathcal{A}$ is an NFA that recognizes $\mathcal{L}$, then $\text{width}(\mathcal{A}) \geq 4$. First, notice that in the minimum DFA (see Figure 4.11) we have $b_i \in O_{q_i}$, for $i = 1, \ldots, 4$, and $\{b_i, b_j\} \not\subseteq O_{q_i} \cap O_{q_j}$, if $i \neq j$. Hence, if $\mathcal{A}$ is an NFA that recognizes $\mathcal{L}$, for all $i = 1, \ldots, 4$ there must exists a state p_i of $\mathcal{A}$ such that a) the language $a_i x^* \cap I_{p_i}$ contains infinitely many strings and b) p_i can reach a final state reading b_i. Since, in the minimum DFA, $\{b_i, b_j\} \not\subseteq O_{q_i} \cap O_{q_j}$ if $i \neq j$, then the output languages of the states $p_1, \ldots, p_4$ must be pairwise distinct, hence the states themselves must be pairwise distinct. Moreover, if $i \neq j$, either $a_i x^* \cap I_{p_j} = \emptyset$ or $a_j x^* \cap I_{p_i} = \emptyset$, otherwise we would have $\{b_i, b_j\} \subseteq O_{q_i} \cap O_{q_j}$ in the minimum DFA. Hence, the sets $I_{p_1}, \ldots, I_{p_4}$ are incomparable with respect to $\preceq_{ps}$. By Lemma 51, the states $p_1, \ldots, p_4$ must be incomparable w.r.t. any colex order on $\mathcal{A}$ and this imply that $\text{width}(\mathcal{A}) \geq 4$.

4.4 Conclusions

In this chapter, we began to explore some of the properties of the width p of NFAs, a measure that allows transferring the nice properties of Wheelerness to any automata, provided p is treated as a constant. Although this measure has been extensively

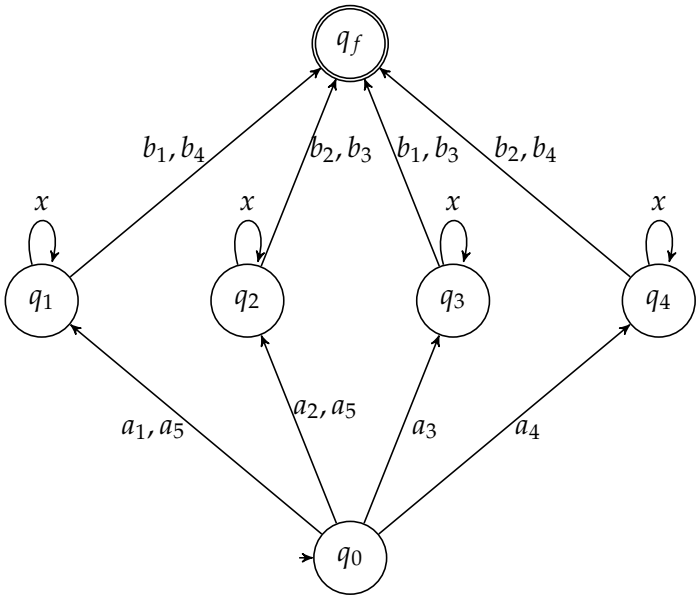


FIGURE 4.12: An NFA recognizing $\mathcal{L}$ of width 4.

studied in [23], many questions about it remain. For example, it has not yet been proved whether determining the non-deterministic width of a given language $\mathcal{L}$ is a decidable problem: while it is very simple to find an upper bound for this value (just take an automaton that recognizes $\mathcal{L}$ and count the number of its states), it is still unclear whether it is possible to define an upper bound for the number of states of an automaton that realizes the minimum width.

Our contribution has been to provide two measures, entanglement$^+$ and entanglement*, which we have proven to be lower bounds to the non-deterministic width of a language $\mathcal{L}$. A positive aspect of these measures is that they are computed on the minimum DFA $\mathcal{D}_\mathcal{L}$. Unfortunately, these lower bounds are not tight. The challenge remains to find a new definition of non-deterministic entanglement, even more accurate than the two we proposed, that can exactly capture the notion of non-deterministic width. Note that if this new measure were computable on the minimum DFA, the calculation of the non-deterministic width would be decidable.

Another possible approach to explore is the following. Theorem 48, which links the entanglement of a language to its deterministic width, was proved in [23] using the definition of the *Hasse* automaton. This automaton 'untangles' the entangled states of the minimum automaton (by appropriately duplicating them), thus achieving the minimum width. This proof sheds light on the complications that arise when trying to untangle entangled states, but unfortunately, it is not constructive. A possible approach to make it constructive could be to use Algorithm 1 described in Chapter 2 to create a partial fingerprint that guides us in constructing the Hasse automaton. However, this approach must be handled carefully, since Algorithm 1 does not terminate when it receives the minimum automaton of a non-Wheeler language as input. Nonetheless, it is plausible that the algorithm can be executed a finite number of times and then stopped once it has provided sufficient information.

Chapter 5

Final remarks and open problems

In this thesis, we explored computational complexity issues related to ordering the states of finite automata. Ordering objects can simplify storage and manipulation tasks significantly. For finite automata, ordering can ease index construction, membership testing, and determinization of NFAs. However, finding the correct order is key and more challenging for NFAs compared to DFAs.

We highlighted the complexity of algorithms when starting with NFAs, proving that several order-related results that guarantee polynomial time algorithms on DFAs become more complex for NFAs, even reduced ones. A key open problem is determining the optimal complexity for an algorithm computing the minimum WDFA from a language fingerprint. Our proposed algorithm is almost optimal except for a factor of n, which is negligible when the output size m is exponential in n. The remaining $\log m$ factor arises from needing binary search to determine the edges of the minimum WDFA, but exploiting specific characteristics of the fingerprint might remove this factor.

We also proved that path-coherent semiautomata admit a linear-sized cascade decomposition into permutation-reset semiautomata, with Wheeler semiautomata having permutation-free cascades. The simplicity of our proofs stems from the order imposed by path coherence and Wheelerness, allowing immediate admissible state decomposition of the initial automaton, satisfying a critical condition of Theorem 41.

Lastly, we examined the width p of NFAs, a measure for extending Wheelerness properties to any automata with p as a constant. Although studied extensively, it remains unclear if determining the non-deterministic width of a language $\mathcal{L}$ is decidable. While finding an upper bound for $\text{width}^{nd}(\mathcal{L})$ is easy, defining an upper bound for the states of an automaton realizing it is not. Our contributions include two measures, entanglement^{+} and entanglement*, as lower bounds for non-deterministic width, computed on the minimum DFA. Despite not being tight, these measures suggest the challenge of defining a more accurate non-deterministic entanglement measure to make the non-deterministic width computation decidable.

Besides those already mentioned, there are other open questions worth discussing. Firstly, it would be valuable to extend the analysis of state complexity on WDFAs to operations beyond the direct product and cascade product. Although WDFAs are not closed under the remaining 'classic' operations, there are some for which they are, namely union, concatenation, and difference with finite languages.

Secondly, there remains an unresolved issue regarding the non-deterministic width of regular languages. In [23] it was shown that for any NFA, every colex order is contained within the *ps* order, which is the order on states induced by the $\prec_{ps}$ relation defined on their input languages. Generally, therefore, the width of the ps order is smaller than the width of any possible colex order, and there are examples where the ps width of an automaton is strictly less than its colex width. However, if we shift our focus to languages and define the ps width of a language $\mathcal{L}$ as the

smallest among the ps widths of the NFAs that recognize it, we cannot find an example where the ps width is strictly less than the non-deterministic width (i.e., the smallest colex width of the NFAs that recognize $\mathcal{L}$). Our educated guess is that, for languages, ps width and non-deterministic width coincide, but this issue remains open at present.

This leaves us with the following open problems:

- Can we extend the analysis of state complexity on WDFAs to operations beyond the direct product and cascade product?
- Can we exploit the characteristics of the fingerprint provided by Algorithm 1 to design an optimal algorithm for constructing the minimum WDFA? Alternatively, can we modify Algorithm 1 so that the fingerprint provided as output can be utilized for this purpose?
- Can we utilize Algorithm 1 to guide us in the construction of the Hasse automaton, or more generally, of any automaton —possibly the smallest one— that achieves the minimum deterministic width for a given language?
- Can we exploit Wheeler KRDT in order to provide a characterization of Wheeler automata as that class of automata decomposable into a cascade of Wheeler reset automata?
- Is there an upper bound for the number of states of an automaton that realizes the minimum non-deterministic width of a language, hence proving that the problem of determining such width is decidable?
- We have provided two lower bounds for the non-deterministic width of a language. Can we refine these lower bounds so that we can exactly capture the notion of non-deterministic width?
- Is it true that the non-deterministic width and the ps width of any regular language coincide?

Bibliography

[1] Janusz A. Brzozowski and Faith E. Fich. "Languages of R-trivial Monoids". In: *J. Comput. Syst. Sci.* 20.1 (1980), pp. 32–49. DOI: 10.1016/0022-0000(80)90003-3. URL: https://doi.org/10.1016/0022-0000(80)90003-3.

[2] Thomas Schwentick, Denis Thérien, and Heribert Vollmer. "Partially-Ordered Two-Way Automata: a New Characterization of DA". In: *Developments in Language Theory, 5th International Conference, DLT 2001, Vienna, Austria, July 16-21, 2001, Revised Papers*. Ed. by Werner Kuich, Grzegorz Rozenberg, and Arto Salomaa. Vol. 2295. Lecture Notes in Computer Science. Springer, 2001, pp. 239–250. DOI: 10.1007/3-540-46011-X_20. URL: https://doi.org/10.1007/3-540-46011-X_20.

[3] Tomás Masopust and Markus Krötzsch. "Partially Ordered Automata and Piecewise Testability". In: *Log. Methods Comput. Sci.* 17.2 (2021). URL: https://lmcs.episciences.org/7475.

[4] H.-J. Shyr and G. Thierrin. "Ordered Automata and Associated Languages". In: *Tamkang J. Math* 5 (1974), pp. 9–20.

[5] Travis Gagie, Giovanni Manzini, and Jouni Sirén. "Wheeler graphs: a framework for BWT-based data structures". In: *Theoretical Computer Science* 698 (2017). Algorithms, Strings and Theoretical Approaches in the Big Data Era (In Honor of the 60th Birthday of Professor Raffaele Giancarlo), pp. 67 –78. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2017.06.016.

[6] Michael Burrows and David J Wheeler. *A block-sorting lossless data compression algorithm*. Tech. rep. 124. Digital Equipment Corporation, 1994.

[7] Giovanni Manzini. "The Burrows-Wheeler transform: theory and practice". In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 1999, pp. 34–47.

[8] Jarno N. Alanko et al. "Tunneling on Wheeler graphs". In: *Data Compression Conference, DCC 2019, Snowbird, UT, USA, March 26-29, 2019*. Ed. by Ali Bilgin et al. IEEE, 2019, pp. 122–131. DOI: 10.1109/DCC.2019.00020. URL: https://doi.org/10.1109/DCC.2019.00020.

[9] Nicola Prezza. "On Locating Paths in Compressed Tries". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. Ed. by Dániel Marx. SIAM, 2021, pp. 744–760. DOI: 10.1137/1.9781611976465.47. URL: https://doi.org/10.1137/1.9781611976465.47.

[10] P. Ferragina and G. Manzini. "Opportunistic data structures with applications". In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. 2000, pp. 390–398. DOI: 10.1109/SFCS.2000.892127.

[11] Nicola Cotumaccio and Nicola Prezza. "On Indexing and Compressing Finite Automata". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. Ed. by Dániel Marx. SIAM, 2021, pp. 2585–2599. DOI: 10.1137/1.9781611976465.153. URL: https://doi.org/10.1137/1.9781611976465.153.

[12] Jarno Alanko, Nicola Cotumaccio, and Nicola Prezza. "Linear-time Minimization of Wheeler DFAs". In: *2022 Data Compression Conference (DCC)*. 2022, pp. 53–62. DOI: 10.1109/DCC52660.2022.00013.

[13] Nicola Cotumaccio. "Graphs can be succinctly indexed for pattern matching in $O(|E|^2 + |V|^{5/2})$ time". In: *2022 Data Compression Conference (DCC)*. 2022, pp. 272–281. DOI: 10.1109/DCC52660.2022.00035.

[14] Alessio Conte et al. "Computing matching statistics on Wheeler DFAs". In: *2023 Data Compression Conference (DCC)*. 2023, pp. 150–159. DOI: 10.1109/DCC55655.2023.00023.

[15] Nicola Cotumaccio et al. "Space-Time Trade-Offs for the LCP Array of Wheeler DFAs". In: *String Processing and Information Retrieval*. Ed. by Franco Maria Nardini, Nadia Pisanti, and Rossano Venturini. Cham: Springer Nature Switzerland, 2023, pp. 143–156. ISBN: 978-3-031-43980-3.

[16] Jarno Alanko et al. "Wheeler Languages". In: *Inf. Comput.* 281.C (Dec. 2021). ISSN: 0890-5401. DOI: 10.1016/j.ic.2021.104820. URL: https://doi.org/10.1016/j.ic.2021.104820.

[17] Daniel Gibney and Sharma V. Thankachan. "On the Hardness and Inapproximability of Recognizing Wheeler Graphs". In: *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*. 2019, 51:1–51:16. DOI: 10.4230/LIPIcs.ESA.2019.51. URL: https://doi.org/10.4230/LIPIcs.ESA.2019.51.

[18] Henning Bordihn, Markus Holzer, and Martin Kutrib. "Determination of finite automata accepting subregular languages". In: *Theoretical Computer Science* 410.35 (2009). Descriptional Complexity of Formal Systems, pp. 3209–3222. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2009.05.019. URL: https://www.sciencedirect.com/science/article/pii/S0304397509003788.

[19] M.P. Schützenberger. "On finite monoids having only trivial subgroups". In: *Information and Control* 8.2 (1965), pp. 190–194. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(65)90108-7. URL: https://www.sciencedirect.com/science/article/pii/S0019995865901087.

[20] J. A. Brzozowski and Rina Cohen. "On Decompositions of Regular Events". In: *J. ACM* 16.1 (1969), 132–144. ISSN: 0004-5411. DOI: 10.1145/321495.321505. URL: https://doi.org/10.1145/321495.321505.

[21] Kai Salomaa and Sheng Yu. "NFA to DFA transformation for finite languages over arbitrary alphabets". In: *J. Autom. Lang. Comb.* 2.3 (1998), 177–186. ISSN: 1430-189X.

[22] Marek Chrobak. "Finite automata and unary languages". In: *Theoretical Computer Science* 47 (1986), pp. 149–158. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(86)90142-8. URL: https://www.sciencedirect.com/science/article/pii/0304397586901428.

[23] Nicola Cotumaccio et al. “Co-lexicographically Ordering Automata and Regular Languages - Part I”. In: *J. ACM* 70.4 (2023), 27:1–27:73. DOI: 10.1145/3607471. URL: https://doi.org/10.1145/3607471.

[24] Kenneth Krohn and John Rhodes. “Complexity of Finite Semigroups”. In: *Annals of Mathematics* 88.1 (1968), pp. 128–160. ISSN: 0003486X. URL: http://www.jstor.org/stable/1970558 (visited on 06/10/2023).

[25] Abraham Ginzburg. *Algebraic theory of automata*. English. Ed. by Academic Press. New York, 1968.

[26] Karl-Heinz Zimmermann. “On Krohn-Rhodes Theory for Semiautomata”. In: *CoRR* abs/2010.16235 (2020). arXiv: 2010.16235. URL: https://arxiv.org/abs/2010.16235.

[27] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN: 0321455363.

[28] Robert McNaughton and Saymour A. Papert. *Counter-Free Automata*. USA: The MIT Press, 1971. ISBN: 9780262130769.

[29] Travis Gagie, Giovanni Manzini, and Jouni Sirén. “Wheeler Graphs: a Framework for BWT-based Data Structures”. In: *Theoretical computer science* 698 (2017), pp. 67–78.

[30] Jarno Alanko et al. “Regular Languages meet Prefix Sorting”. In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*. 2020, pp. 911–930. DOI: 10.1137/1.9781611975994.55. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611975994.55. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611975994.55.

[31] Jaroslav Opatrny. “Total Ordering Problem”. In: *SIAM J. Comput.* 8.1 (1979), pp. 111–114. DOI: 10.1137/0208008. URL: https://doi.org/10.1137/0208008.

[32] Jarno Alanko et al. “Wheeler languages”. In: *Inf. Comput.* 281 (2021), p. 104820. DOI: 10.1016/j.ic.2021.104820. URL: https://doi.org/10.1016/j.ic.2021.104820.

[33] Ruben Becker et al. “Optimal Wheeler Language Recognition”. In: *String Processing and Information Retrieval: 30th International Symposium, SPIRE 2023, Pisa, Italy, September 26–28, 2023, Proceedings*. Pisa, Italy: Springer-Verlag, 2023, 62–74. ISBN: 978-3-031-43979-7. DOI: 10.1007/978-3-031-43980-3_6. URL: https://doi.org/10.1007/978-3-031-43980-3_6.

[34] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. “The State Complexities of Some Basic Operations on Regular Languages”. In: *Theor. Comput. Sci.* 125 (1994), pp. 315–328.

[35] Giovanni Manzini et al. “The *rational* construction of a Wheeler DFA”. To appear in proceedings of CPM 2024. 2024.

[36] John Hopcroft. “An n log n Algorithm for Minimizing States in a Finite Automaton”. In: *Theory of Machines and Computations*. Ed. by Zvi Kohavi and Azaria Paz. Academic Press, 1971, pp. 189–196. ISBN: 978-0-12-417750-5. DOI: https://doi.org/10.1016/B978-0-12-417750-5.50022-1. URL: https://www.sciencedirect.com/science/article/pii/B9780124177505500221.

[37] Walter J. Savitch. "Relationships between nondeterministic and deterministic tape complexities". In: *Journal of Computer and System Sciences* 4.2 (1970), pp. 177–192. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(70)80006-X. URL: https://www.sciencedirect.com/science/article/pii/S002200007080006X.

[38] Oded Maler and Amir Pnueli. "Tight Bounds on the Complexity of Cascaded Decomposition of Automata". In: *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*. IEEE Computer Society, 1990, pp. 672–682. DOI: 10.1109/FSCS.1990.89589.

[39] Attila Egri-Nagy and Chrystopher Nehaniv. "Algebraic Hierarchical Decomposition of Finite State Automata: Comparison of Implementations for Krohn-Rhodes Theory". In: vol. 3317. July 2004, pp. 315–316. ISBN: 978-3-540-24318-2. DOI: 10.1007/978-3-540-30500-2_32.

[40] John Rhodes. *Applications of automata theory and algebra: via the mathematical theory of complexity to biology, physics, psychology, philosophy, and games*. English. Ed. by C.L. Nehaniv. Singapore: World Scientific Publishing, 2010. ISBN: 9812836977.

[41] Dominique Perrin and Jean-Eric Pin. "Infinite Words: Automata, Semigroups, Logic and Games". In: *Bulletin of Symbolic Logic* 11.2 (2005), pp. 246–247.

[42] R. P. Dilworth. "A Decomposition Theorem for Partially Ordered Sets". In: *Annals of Mathematics* 51.1 (1950), pp. 161–166. ISSN: 0003486X. URL: http://www.jstor.org/stable/1969503.